”På endast 186 sidor lyckas Dan Josefsson med det omöjliga: att enkelt, begripligt, ingående och ändå heltäckande beskriva hur du skapar nätsidor. En utmärkt utbildningsbok.”

SYDSVENSKA DAGBLADETS HERMOD PEDERSEN OM VERSION 1.0

FLER RÖSTER PÅ INSIDAN

NYCKELN TILL DIN EGEN

hemsida på Internet

DAN JOSEFSSON • VERSION 3.1 • 2004

NY REVIDERAD UTGÅVA!

Inkluderar specialkoder för Internet Explorer och Netscape och introduktion till Style sheets och JavaScript.

PRISMA

Sedan första upplagan av Nyckeln till din egen hemsida kom ut har boken blivit det självklara valet för journalistutbildningar, studiecirklar, datorkurser och inte minst för tusentals hemmapraktiker som vill lära sig göra professionella hemsidor.

Läsare och kritiker är helt enkelt överens om att boken fyller ett stort tomrum i den svenska datorlitteraturen. Äntligen en välskriven handbok som på ett enkelt och pedagogiskt sätt berättar hur det går till att göra hemsidor.

Utan allt det där onödiga frossandet i tekniska termer, men ändå utan att utelämna någonting. Genom att börja från grunden och sedan följa läsaren hela vägen fram till den allra mest avancerade HTML-programmeringen är boken ovärderlig både för dig som aldrig någonsin gjort en hemsida och för dig som redan hunnit en bit på väg.

Boken innehåller över 100 förklarande bilder och exempel. Bokens egen hemsida (www.etc.se/nyckeln) innehåller förutom samtliga exempel också en mängd tips och trick som gör din egen hemsida bättre och vackrare.

Detta är den tredje, reviderade upplagan av Nyckeln till din egen hemsida på Internet. Bland nyheterna märks specialkoderna för Internet Explorer 4.0 och Netscape Navigator 4.0. Dessutom ingår ett nyskrivet och utökat kapitel om Style sheets samt en helt ny introduktion till JavaScript.

Pressröster om version 1.0 av Nyckeln till din egen hemsida på Internet:

”Författaren är en lysande pedagog med ett sjätte sinne för eleven/läsarens behov.”

BJÖRN SJÖ, ARBETET NYHETERNA

”Det här är den bästa boken jag sett i kategorin ’nybörjarguide till egen hemsida’. Den är skriven på ett sådant sätt att inga förkunskaper krävs, varken i HTML, Internet eller om datorer.”

INTERNET WORLD

”Nyckeln till din egen hemsida på Internet är det bästa jag läst på svenska i det här facket. I motsats till de snabbproducerade pocketlättviktarna om Internet från Pagina förlag är detta en guldgruva att ösa ur både för nybörjaren och för den som har en del erfarenhet i konsten att skapa en hemsida.”

PETER FRÖBERG, NORRKÖPINGS TIDNINGAR

OM FÖRFATTAREN:
Dan Josefsson är tidnings- och TV-journalist och har skrivit om IT i bland annat Internet Guiden och Aftonbladet. Sedan 1995 ägnar han sig åt grävande journalistik på tidningen ETC, bland annat som redaktör för tidningens Internetupplaga Dagens ETC.

DAN JOSEFSSON • VERSION 3.1 • 2004

NYCKELN TILL DIN EGEN HEMSIDA PA INTERNET

Innehåll

	Inledning	**1**
Kapitel 1	**Hemsidornas hem: World Wide Web** World Wide Web slår igenom 4 Vad är en hemsida? 5	**3**
Kapitel 2	**Innan vi sätter igång: Utrustningen** Att välja webläsare 9 Arbetsmetodik 13	**7**
Kapitel 3	**Din första hemsida: Grunderna** Hemsidans beståndsdelar 18 Gör ett malldokument 20 Att formatera text i HTML 22 Olika sätt att dela upp texten 24 Att ändra storlek på text 28 Koderna i sammanfattning 35	**17**
Kapitel 4	**Att koppla ihop sidorna: Hyperlänkar** Relativa sökvägar 40 Absoluta sökvägar 43 Länkar inom ett dokument 44 Att skicka brev från hemsidan 47 Länkar i nya fönster 47 Koderna i sammanfattning 48	**37**
Kapitel 5	**Organisera din hemsida: Listor i HTML** Koderna i sammanfattning 54	**49**
Kapitel 6	**Ge liv åt din hemsida: Bilder** Olika typer av bilder 56 Att justera bilder 60 Bilder som länkar 63 Bildstorlekar 66 Alternativtexter 67 Bakgrundsbilder 68 Transparenta Gif-bilder 69 Koderna i sammanfattning 70	**55**
Kapitel 7	**Flera länkar i en bild: Bildkartor** Serverbaserade bildkartor 73 Klientbaserade bildkartor 79 Falska bildkartor 82 Koderna i sammanfattning 83	**71**
Kapitel 8	**Styr hemsidans färg: Färgkoder** Koderna i sammanfattning 88	**85**

Kapitel 9 **Styr din hemsidas form: Tabeller** **89**
Att styra tabellens storlek 96
Att justera tabellens innehåll 98
Ramar och luft 100
Bilder och färg i tabellen 102
Koderna i sammanfattning 108

Kapitel 10 **Visa flera sidor på en gång: Rutsystem** **111**
Skapa markeringsdokument 113
Kolumner 113
Rader 115
Den magiska asterisken 116
Länkar mellan rutor 120
Rutsystem utan ramar 122
Om inte webläsaren klarar rutor 124
Koderna i sammanfattning 125

Kapitel 11 **För interaktiva hemsidor: Formulär** **127**
Formulärets uppbyggnad 128
Formulärobjekten 130
Koderna i sammanfattning 134

Kapitel 12 **När sidan ska ut på nätet: Uppladdning** **135**

Kapitel 13 **Webläsarens dolda stöd: CGI-script** **139**

Kapitel 14 **Full kontroll över sidans form: Style sheets** **141**
Lokala style sheets 143
Globala style sheets 144
Klasser 146
ID 147
Bild och text i lager 153
Länkade style sheets 155
Dynamiska typsnitt 156
Koderna i sammanfattning 170

Kapitel 15 **Ett steg bortom HTML: Dynamisk HTML** **159**
JavaScript 160
Objektmodellen 163
Properties och methods 164
Event handlers 166
De konstiga citattecknen 167

Kapitel 16 **Koder för särskilda läsare: Internet Explorer** **171**
Rullande text 172
Flytande rutor 173
Nya marginaler och fixerad bakgrund 174
Bakgrundsljud 174
Bestäm färg på tabellramar 175
En ny tabellorganisation 175
Koderna i sammanfattning 179

Kapitel 17 **Koder för särskilda läsare: Netscape** **181**
Visa text i flera spalter 181
En kod som skapar luft 182
Jobba i lager 184
Koderna i sammanfattning 186

Kapitel 18 **Konsten att visa specialtecken: Teckenkoder** **187**

Bilaga 1 **HTML-referens** **191**

Bilaga 2 **Style sheets referens** **200**

Bilaga 3 **Alla teckenkoder** **202**

Bilaga 4 **Lathund för rutsystem** **204**

Register **Sakregister** **207**

Inledning

När media talar om World Wide Web handlar artiklarna och inslagen ofta om hur stora mediaföretag experimenterar med digital publicering. Det är Aftonbladets hemsidor som vinner priser och det är Posten och Telia som tävlar om att bygga den största kommersiella marknadsplatsen för landets växande skara uppkopplade.

Men det är inte mediaföretagen som gjort World Wide Web till vad det är idag. Det är istället de hundratusentals privatpersoner och studenter världen över som skapar egna hemsidor och därmed fyller World Wide Web med personligt innehåll. Resultatet är en fascinerande blandning av högt och lågt. Ursinnig politisk agitation ligger sida vid sida med receptsamlingar och bilderna på familjen Johanssons katt ett musklick från litterära essäer.

Jag vet inte vad just din hemsida ska innehålla. Men en sak vet jag: När du skriver hemsidan skriver du mediahistoria på samma gång. Aldrig förr har människor kunnat masskommunicera utan att vara beroende av oerhört dyra media som TV och radio. Från din egen hemsida utgår därför inget mindre än en kommunikationsrevolution.

Du behöver inga som helst förkunskaper för att använda *Nyckeln till din egen hemsida på Internet*. Boken börjar med de allra mest grundläggande kunskaperna i HTML (språket som används för att skapa hemsidor) för att sedan successivt bli mer avancerad. När du är färdig med kapitel 18 kommer du att behärska den mest avancerade HTML-kodning som finns på World Wide Web idag. Även du som redan provat på HTML och vill lära dig de mer avancerade teknikerna och tricken kommer därför att ha stort utbyte av boken.

Vem är boken skriven för?

Hur det hela är organiserat

De första två kapitlen är främst riktade till dig som aldrig tidigare använt HTML. I kapitel 1 ges en kort tillbakablick på World Wide Webs historia och en genomgång av hur det går till när en hemsida skickas över Internet. Kapitel 2 ger grundläggande kunskaper om HTML-språkets uppbyggnad och tips om program som används för att göra hemsidor. Även du som redan provat på att göra hemsidor kan ha utbyte av avsnittet om att välja rätt koder (sid 11).

I kapitel 3 börjar själva HTML-kodandet. Efter att ha läst kapitlet och gjort övningsexemplen kommer du att kunna tillräckligt för att göra en enkel men komplett hemsida. Du som har erfarenhet av HTML kanske nöjer dig med att titta på avsnittet om malldokument (sid 20).

I kapitel 4–11 går vi igenom en rad olika områden inom HTML-språket. För dig som är nybörjare är det antagligen bäst att fortsätta läsa kapitlen i tur och ordning. Du som har viss vana kan istället välja att hoppa till de avsnitt du behöver lära dig mer om. Lägg särskilt märke till de ingående beskrivningarna av bildkartor, tabeller och rutor (*frames*) i kapitel 7, 9 och 10.

Kapitel 12 visar hur det går till att ladda upp din hemsida så att den kan nås från övriga World Wide Web. Kapitel 14 är helt nyskrivet till den andra upplagan. Där behandlas grundligt den spännande *style sheets*-tekniken,Skicka 1 med vilken du exakt kan styra textstorlek, typsnitt och andra delar av din hemsidas form.

Kapitel 15 ger en introduktion till dynamisk HTML, och motorn som driver denna nya teknik: JavaScript.

Kapitel 16 och 17 handlar om de specialkoder som bara webläsarna Microsoft Internet Explorer respektive Netscape Navigator förstår.

Till sist, i kapitel 18, tittar vi närmare på problemet med svenska tecken, något framför allt du som använder Macintosh behöver tänka på.

I bokens bilagor finner du en HTML-referens, en referens över style sheet-koderna, en lista över samtliga teckenkoder och en lathund som hjälper dig att göra rutsystem.

Missa inte bokens egen hemsida (**http://www.etc.se/nyckeln/**). Där finner du alla bokens kodexempel och annan hjälp som underlättar arbetet med din egen webplats.

Lycka till med din egen hemsida på Internet!

Dan Josefsson
Mars 1998

1 HEMSIDORNAS HEM: World Wide Web

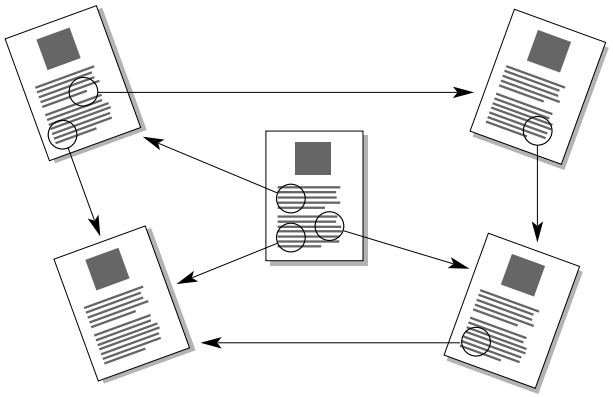
För bara några år sedan var Internet ett okänt begrepp utanför forskarvärlden. Sedan kom World Wide Web och gjorde i ett slag det tidigare så svåranvända Internet lättillgängligt.

Idag strömmar nya användare till i hundratusental, och forskarna som tidigare var ensamma på nätet är nu i minoritet. I det här kapitlet bekantar vi oss närmare med hemsidornas hem – World Wide Web.

Trots att Internet utvecklats oavbrutet sedan 60-talet var det till för bara några år sedan nästan omöjligt för vanliga människor att utnyttja nätet. Redskapen och programmen som krävdes var så komplicerade och svårarbetade att den som inte hade datorer som yrke knappast ens förstod vad Internet var för något. Och även om den lilla skara forskare som hade tillgång till Internet var oerhört entusiastiska över den nya kommunikationsformen, förblev nätet deras egen hemlighet.

Men 1989 började något hända. Forskaren Tim Berners-Lee vid Cern, ett institut för partikelfysik i Genève, tyckte att för mycket tid på institutet gick åt till att leta efter bortkomna forskningsrapporter, programdokumentationer och annan information. Han föreslog ett nytt datorbaserat system för informationslagring, baserat på en idé som funnits sedan 50-talet: hypertext.

Hypertext är en så kallad icke-linjär textform. Det betyder att hypertexter inte alltid är avsedda att läsas från första bokstaven till sista. Istället kan läsaren välja att följa så kallade *länkar* till fördjupande information om särskilda ord eller ämnen som förekommer i texten. Den stora poängen är att länken kan leda till information som ligger på en helt annan dator än det ursprungliga dokumentet (figur 1.1 på nästa sida). Berners-



Figur 1.1
Hypertext är icke-linjär. Ett antal länkar inlagda i texten ger läsaren möjlighet att hoppa till andra dokument. Länkdokumentet behöver inte ligga på samma dator, eller ens i samma land, som startdokumentet.

Kärt barn. World Wide Web har många namn. Vanligt är förkortningen WWW, men också W3 förekommer. Ibland kallas WWW även för "nätet", men den beteckningen betyder oftare Internet som helhet.
Den språkligt minst korrekta termen för WWW torde vara "webben" (som i: "jag surfade på webben 38 timmar i sträck") efter engelskans "the web". En hemsida kallas ibland även för "webbsida".

Lee menade att användandet av hypertext skulle kunna göra stora mängder kunskap tillgängliga och överskådliga, trots att de egentligen var spridda på många olika datorer, som låg i olika länder.

Cerns ledning nappade på projektet, som ett år senare döptes till World Wide Web. Första steget mot en fullständig förvandling av Internet hade därmed tagits.

World Wide Web slår igenom

World Wide Web, eller WWW som det kom att kallas, innebar två stora nyheter för Internetvärlden. För det första var hypertexten i sig ett språng framåt; ett stort problem med Internet hade nämligen alltid varit att hitta i den väldiga informationsmängden. Hypertextlänkarna gav en helt ny möjlighet att samla relaterad information på lätt åtkomliga hemsidor.

För det andra utnyttjade WWW multimedia. Ljud och bild kunde på ett lika självklart sätt som text ingå i dokumenten. World Wide Web gjorde därmed Internet betydligt roligare att använda, något som bidrog till den makalösa utveckling som snart skulle starta.

I ett par år levde det nyfödda World Wide Web ett tynande liv. I början av 1993 fanns det till exempel fortfarande bara 50 webservrar (särskilda datorer som förmedlar hypertextdokument) på hela Internet. Problemet var att det krävdes ett särskilt program för att läsa World Wide Webs hypertextdokument, och att de modeller som fanns av programmet var dåliga och svåranvända.

I utvecklingsarbetet med World Wide Web samarbetade Cern med National Center for Supercomputing Applications vid universitetet i Illinois i USA (NCSA). Det blev NCSA som utvecklade den första riktigt användarvänliga läsaren av hypertextdokument. Den hette Mosaic och fick World Wide Webs utveckling att fullständigt explodera.

Webserver. En webserver (eller www-server) är en dator som är ansluten till Internet och som har till uppgift att skicka hemsidor, bilder och ljud till din webläsare. Mer om detta nedan.

Under 1993 och 1994 växte antalet datorer som ingår i WWW explosionsartat. Ett år senare hade media hakat på Internettrenden, och World Wide Web blev ett begrepp även bland den majoritet svenskar som aldrig ägt en dator. I januari 1996 fanns det 100 000 webservrar i världen, och sedan dess har antalet fördubblats varje halvår. Sommaren 1997 var antalet uppe i 1 203 000. Ingenting tyder på att tillväxttakten håller på att minska.

Vad är en hemsida?

World Wide Web består alltså idag av nästan en och en halv miljon så kallade webservrar. Webservrar är särskilda datorer som har till uppgift att skicka de olika beståndsdelar som ingår i en hemsida kors och tvärs över Internet. Beståndsdelarna kan vara allt från bilder och ljud till animationer, men en komponent måste alltid finnas med: HTML-dokumentet.

Ett HTML-dokument är ett vanligt textdokument som skrivits med sidbeskrivningsspråket HTML (*Hyper Text Markup Language*). Syftet med HTML-dokumentet är att beskriva för en webläsare, till exempel Mosaic eller Netscape, hur hemsidan ska se ut.

En webläsare har två uppgifter: 1) att hämta ett HTML-dokument på en angiven adress inom World Wide Web och 2) att visa sin tolkning av detta HTML-dokument på datorskärmen. För att demonstrera hur processen går till ska vi hämta en hemsida om tibetansk konst från en dator i USA. **Adressen** är **http://www.chron.com/mandala/index.html**. Det går att utläsa en hel del bara genom att titta på adressen (se vänster marginal).

Adressen. Du kan utläsa en hel del av en hemsidas adress. Adressen i exemplet till höger ger förljande information:
http: Hemsidan kommer att skickas med en särskild teknik för överföring av hypertextdokument.
www: Hemsidan finns på World Wide Web.
chron: Sidan finns på en domän som heter chron.
com: Domänen ingår i Internets kommersiella del i USA.
mandala: Hemsidan ligger i en underkatalog som heter mandala.
index.html: Namnet på HTML-dokumentet är index.html.

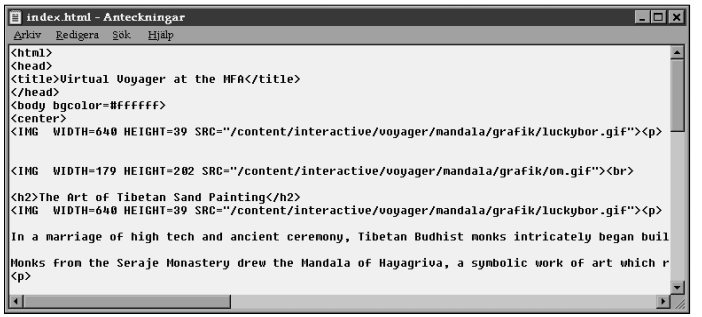
En adress kallas också för **URL**, vilket står för **Uniform Resource Locator**.

Efter att ha studerat adressen knappar vi in den i webläsarens adressfönster och trycker på <RETURN>. I statusfönstret (brukar finnas i webläsarens nederkant) dyker det nu upp ett meddelande som lyder ungefär "Contacting host www.chron.com".

Det betyder att läsaren söker kontakt med den webserver på vars hårddisk dokumentet "index.html" ligger. Efter en stund dyker ett nytt meddelande upp i webläsarens underkant: "Host contacted, waiting for reply". Exakt hur meddelandet är utformat varierar mellan olika läsare, men principen är densamma: läsaren har kontaktat datorn i USA och inväntar svar.

Nu går allt förhoppningsvis mycket fort. Datorn i USA skickar iväg dokumentet "index.html" och ett ögonblick senare har det via Internet nått din dator. Figur 1.2 visar hur "index.html" ser ut. Bli inte förskräckt om

Figur 1.2
En hemsida om tibetansk konst så som din webläsare ser den. I dokumentet finns all text som ingår i hemsidan, men även information om var på Internet bilderna finns att hämta, och var och hur dessa ska placeras i förhållande till texten. Bli inte förskräckt om du tycker att det ser rörigt ut. Det är enklara än man kan tro, och snart kommer du att ha skrivit din första egna hemsida.



du tycker att det verkar krångligt. Det är enklare än det ser ut och du kommer snart själv att skriva betydligt mer avancerade HTML-dokument än det här.

Nu måste webläsaren tolka informationen i index.html. Texten placeras omedelbart ut på skärmen. Bildernas storlek och placering framgår också av HTML-dokumentet, och webläsaren markerar dem därför med tomma rutor. Själva bilderna finns aldrig med i själva HTML-dokumentet, bara information om hur stora de är, om de ska ha ram eller inte och var på Internet de finns att hämta. När webläsarens tolkning av HTML-dokumentet är klar ser hemsidan ut som i figur 1.3.

Sista momentet är att webläsaren hämtar bilderna. Dessa ligger oftast på samma dator som HTML-dokumentet, men de kan också ligga någon annanstans på Internet. Det tar en stund för bilderna att färdas från USA, men slutligen är alla bilderna nedladdade, och hemsidan är färdig (figur 1.4).

Figur 1.3
Texten visas direkt, men bilderna markeras bara med tomma rutor. Nästa steg är att hämta bilderna på de adresser som finns angivna i HTML-dokumentet.

Figur 1.4
När bilderna laddats ned fylls de tomma rutorna och hemsidan är färdig.

Figur 1.3

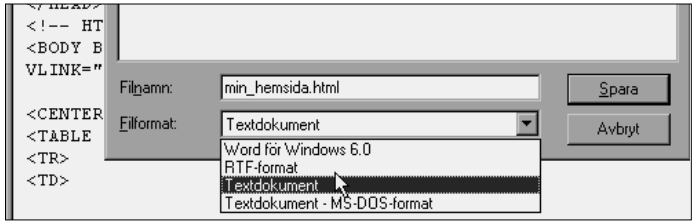
Figur 1.4

2 Utrustningen

Det här kapitlet handlar om de enkla hjälpmedel du behöver för att göra din hemsida. Vi tittar också på de vanligaste webläsarna och på hur konkurrensen dem emellan påverkar din hemsidas utformning. Till sist mjukstartar vi med HTML-kodning och ser hur arbetet med en hemsida bäst läggs upp.

I förra kapitlet konstaterade vi att alla hemsidor egentligen är helt vanliga textdokument, skrivna med sidbeskrivningsspråket HTML, *Hyper Text Markup Language*. Du som använder **PC** kan skriva dina hemsidor med en så kallad texteditor (figur 2.1), till exempel ”Anteckningar” i Windows. Det går också bra att använda ett vanligt ordbehandlingsprogram, till exempel Word. Men då måste du komma ihåg att alltid spara dina dokument i rent textformat. Ordbehandlingsprogram lägger nämligen normalt information om typsnitt, formatering, marginaler och liknande i dokumen-

Figur 2.1
En vanlig texteditor duger bra för att skriva hemsidor. Om du använder ett fullfjädrat ordbehandlingsprogram måste du vara noga med att spara dokumentet som ren text. Annars kommer hemsidan inte att fungera.



Figur 2.2
Om du skriver HTML med hjälp av ett ordbehandlingsprogram är det viktigt att välja "Textdokument" eller "Enbart text" (beteckningen varierar beroende på program) som filformat då dokumentet sparas. Annars kommer HTML-dokumentet inte att fungera.

ten. Denna information kommer att förstöra din hemsida. I figur 2.2 ovan ser du hur det går till att spara dokumentet som ren text i Word.

För dig som använder Macintosh är det inte riktigt lika enkelt. Macintoshar klarar nämligen inte att skriva svenska tecken (å, ä och ö) så att de visas rätt på hemsidor. Som Macägare måste du därför ordna en så kallad HTML-editor.

HTML-editorer – ett måste för Macägare

De flesta HTML-editorer är i princip vanliga texteditorer som försetts med vissa specialfunktioner för att förenkla skrivandet av HTML-kod. För dig som använder Macintosh är en av dessa specialfunktioner nästan nödvändig: automatisk översättning av alla svenska tecken till särskilda koder som webbläsaren förstår. Stora "Ä" måste till exempel skrivas "&Aouml;" och "å" "å" för att det ska bli rätt. Detta kan du naturligtvis göra för hand, men roligt är det inte.

En bra editor som automatiskt översätter svenska tecken till specialkoder är "HTML Editor", som du hittar på: http://dragon.acadiau.ca/~giles/. Programmet är **shareware** och är väl värt att registrera. Observera att detta **bara gäller Macägare**. Du som skriver HTML-kod på PC kan skriva svenska tecken precis som vanligt. Mer om varför just Macintosh har så svårt för svenska tecken hittar du i kapitel 18.

En mer avancerad HTML-editor för Macintosh som majoriteten av de professionella hemsidesbyggarna använder heter **BBEdit**. En gratis provversion laddar du ned på adress http://www.bbedit.com.

Även om PC-ägare inte behöver specialtecken och därför klarar sig långt med en gratis texteditor som "Anteckningar", kan en HTML-editor förenkla arbetet, särskilt om du planerar att göra många sidor. En bra HTML-editor för Windows heter **HTML-Kit** och du finner en gratisversion på adress **http://www.chami.com/html-kit/download/**.

Ett tips till dig som använder PC är att börja göra bokens övningsuppgifter med "Anteckningar" och sedan ladda ned HTML-Kit när du fått blodad tand på HTML-kodning.

Shareware.
Shareware-systemet bygger på att du som användare får prova ett program under en viss tid, ofta 30 dagar. Om du sedan vill fortsätta att använda programmet måste du betala för det. Detta kallas ofta att *registrera* programmet hos tillverkaren.

Att välja webbläsare

Sedan World Wide Web började utvecklas på Cern i slutet av 80-talet har antalet webbläsare, det vill säga de program som läser HTML-dokument, ökat explosionsartat. Mosaic, programmet som startade alltihop, uppdateras inte längre och har nästan helt försvunnit. I dess ställe har Netscape Navigator och Microsoft Internet Explorer nästan fullständigt kommit att dominera marknaden. Det är därför klokt att välja en av dessa läsare när du vill titta på dina hemsidor under utvecklingsarbetet. Valet måste göras med omsorg. För att förstå varför ska vi titta på vad som hänt på World Wide Web de senaste åren.

HTML 2.0 – den första standarden

Som du kommer att se lite längre fram i det här kapitlet består HTML-språket av en mängd koder som talar om för webbläsaren hur en hemsida ska se ut. En kod talar till exempel om att ett visst textavsnitt ska visas med fet stil, medan en annan mer avancerad kod får texten att röra sig fram och tillbaka i webbläsarens fönster.

Tanken med World Wide Web har hela tiden varit att HTML-språket ska vara standardiserat, så att en kod alltid tolkas på samma sätt oavsett vilken webbläsare som används. Idag är det *The World Wide Web Consortium* (förkortat W3C) som har hand om standardiseringsarbetet. W3C är ett samarbete mellan många olika företag som utvecklar Internetprodukter. Verksamheten drivs dels av Laboratory for Computer Science vid Massachusetts Institute of Technology (MIT) i USA, dels av INRIA, ett franskt institut för datorvetenskap.

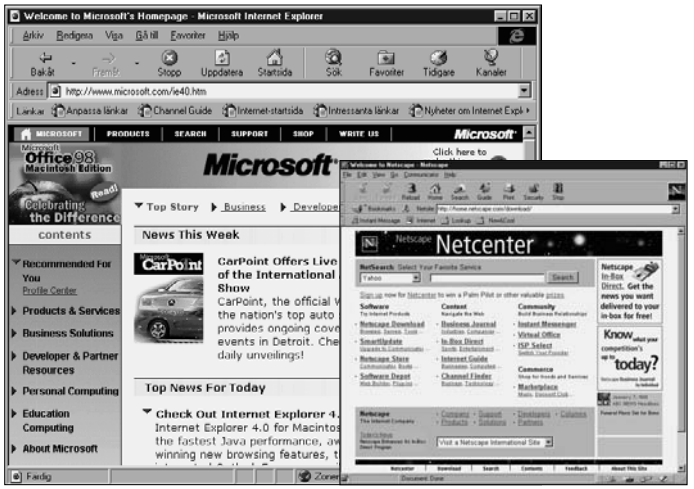
Den första standardiserade **versionen** av HTML hette HTML 2.0. Att namnet inte blev HTML 1.0, vilket ju hade varit logiskt, beror på att man ville skilja den nya versionen från den mer informella variant av HTML som dittills använts.

Netscape Navigator gör entré

Den webbläsare som först fick allmän spridning var Mosaic, som utvecklades av en grupp programmerare vid NCSA i USA. Eftersom NCSA också var med om att utveckla hela World Wide Web tillsammans med Cern i Genève, stödde Mosaic W3Cs HTML-standard till punkt och pricka.

Men i gruppen av programmerare som utvecklade Mosaic arbetade en man vid namn Marc Andreessen. Han insåg snart den enorma potentialen i World Wide Web och startade därför företaget Netscape, som 1994 släppte en egen webbläsare: Netscape Navigator. På kort tid tog Netscape över

Versioner.
När ett datorprogram utvecklas brukar man ge det olika versionsnummer allteftersom arbetet fortskrider. Om numret är under 1 (t ex 0.89) är programmet ännu under utveckling, och fungerar alltså inte riktigt som det ska. En sådan version kallas för **betaversion**. När programmet är färdigt för försäljning kallas det normalt för version 1.0. Små förändringar ger därefter nya tiondelar (t ex 1.1, 1.2 osv). Om programmet radikalt omarbetats och fått nya funktioner ändras heltalen: 2.0. Ett långlivat program som Word för Windows är vid det här laget uppe i version 7.0.



Figur 2.3
Webbläsare. Netscape Navigator och Microsoft Internet Explorer är de överlägset mest använda läsarna. Här ses version 4.0 av båda programmen.

en mycket stor del av marknaden för webbläsare, vilket berodde på att Netscapes läsare var bättre än Mosaic, men ändå gratis.

Men Netscape Navigator höll sig inte alls till den HTML-standard som W3C tagit fram. Istället införde man en lång rad nya HTML-koder som bara Netscape Navigator förstod. Många inom Internetvärlden upplevde detta som ett brott mot hela World Wide Webs huvudpoäng, som ju är att information ska kunna nås oavsett var på jorden man befinner sig och vilken dator och programvara man använder. Netscape, som själv är medlem av W3C, lät sig dock inte bekomma av kritiken, utan erbjöd istället W3C att ta med Netscapes nya koder i HTML 3.0, som var under utveckling.

Microsoft Internet Explorer gör entré

Medan Netscape inte bara spred sin webbläsare till alla världens World Wide Web-användare, utan dessutom tog över initiativet i utvecklingen av hela HTML-språket, gjorde mjukvarujätten Microsoft till en början ingenting alls. Microsoft förstod helt enkelt inte Internets storhet. 1995 hade företagets ledare Bill Gates dock insett att Microsoft helt höll på att missa den viktigaste utvecklingen på datorområdet sedan persondatorn uppfanns, och Microsoft släppte därför en egen webbläsare: Internet Explorer.

Utmärkande för Internet Explorer version 1.0 var att den förstod nästan alla Netscapes specialkoder. Tanken var uppenbarligen att hemsidor som gjorts för att ses med Netscape Navigator också skulle kunna ses med Internet Explorer. Därmed stod det också klart att Microsoft ämnade ta upp kampen med Netscape om herraväldet på World Wide Web.

Tips!
Läs mer om försöken att standardisera HTML på W3Cs egen hemsida. Adressen är: http://www.w3.org/

HTML-standarder idag

Sedan Microsoft Internet Explorer 1.0 släpptes har mycket hänt. Bill Gates gav Netscape ett hårt slag genom att deklarera att Internet Explorer ska förbli ett gratisprogram. Nya versioner har släppts i rask takt, och Internet Explorer 4.0 innehöll förutom nästan alla Netscapes specialkoder också en rad egna varianter som inte en nyare version av Netscape (4.0) förstod. I januari 1998 slog Netscape tillbaka genom att göra även sin webbläsare till ett gratisprogram.

W3C tvingades å sin sida ge upp arbetet med HTML 3.0, eftersom standarden blev omsprungen av Netscape och Microsoft innan den ens var färdig. Istället försökte man ta med Netscapes nya koder i en ny standard, HTML 3.2. Innan man var klar hade både Netscape och Internet Explorer återigen skapat nya koder, men idag har W3C faktiskt slutfört arbetet med en HTML-standard som man kallar HTML 4.0. Både Netscape och Microsoft lovade att i kommande modeller av sina webbläsare stödja denna standard.

Den senaste HTML-standarden som W3C utarbetat kallas HTML 4.01 och blev klar 1998. Dessvärre håller sig varken Microsoft eller Netscape helt och hållet till denna standard vilket innebär att vissa koder och specificeringar - framförallt när det gäller style sheets - fungerar lite olika med olika typer av webbläsare. För att göra det hela ännu lite mer irriterande finns det också skillnader mellan de olika webbläsarnas operativsystemsversioner. Något som fungerar bra på Internet Explorer för PC kanske inte ser alls så bra ut på Internet Explorer för Mac. Dessutom finns det förstås skillnader mellan webbläsarnas versioner.

Hur kommer man nu tillrätta med det här? Jo, det finns förstås sätt att ta sig an problemen. Först och främst är det mycket viktigt att du är klar över vem du riktar dig till innan du börjar arbetet med att koda dina webbsidor. Ju snävare målgrupp du har desto lättare är det att ta reda på vilken utrustning och vilka program målgruppen använder. När du har bildat en uppfattning om de presumtiva användarnas förutsättningar kan du lättare avgöra hur du ska utforma dina sidor, både i fråga om visuell och kodmässig design.

Alltså, bestäm först vad du vill åstadkomma, försök att undvika lösningar och koder som tolkas olika på olika webbläsare. Om du har tillgång till det, titta på dina sidor i så många olika webbläsare och miljöer som möjligt. Försök hitta "enkla" lösningar, enkelhet behöver inte innebära "tråkighet" och begränsningar utan den enkla lösningen kan ibland vara den intelligenta och snygga lösningen.

Att välja rätt koder

Ända sedan Netscape började hitta på egna HTML-koder som bara fungerade om man såg hemsidan med just deras webbläsare, har den stora frågan varit vilka koder man egentligen ska använda.

För att hjälpa dig att svara på den frågan avslutas de flesta kapitlen i den här boken med en sammanfattning av alla koder som nämnts i kapitlet. Så här kan en sammanfattning se ut:

Om det står ”alla” till höger betyder det att koden förstås av i princip

HTML-kod	Webbläsare
<html>…</html>	alla
<blink>…</blink>	ns2-4
<marquee>…</marquee>	ie3-4
<style>…</style>	ie3-4·ns4

alla webbläsare som används idag.

”ns2-4” betyder Netscape Navigator version 2, 3 och 4 och ”ie3-4·ns4” betyder att koden fungerar med Internet Explorer version 3 och 4 och uppåt, samt Netscape Navigator version 4 och uppåt.

Så vilka koder ska just du använda?

Det beror på vem du vänder dig till. Man kan som vägledning dela in dem som besöker din hemsida i tre grupper.

Om du håller dig till koder som alla läsare förstår, kommer din hemsida

En sida för alla

att kunna ses med nästan alla webbläsare som finns. Det innebär att den blir maximalt tillgänglig och exempelvis kan ses även av folk som envisas med att använda antika webbläsare som Mosaic, mobiltelefoner med surfmöjlighet eller äldre datorer som Amiga, Atari eller till och med Commodore 64. Den här graden av tillgänglighet innebär att du måste avstå från finesser som rutor (*frames*), olikfärgad text och mycket annat. Rekommenderas bara till dig som verkligen inte vill stänga ute någon.

En annan strategi är att använda koder som både Netscape 6.x och Internet

En sida för de flesta

Explorer 5.x förstår. Dessa båda webbläsare är så spridda att minst 95 procent av alla som använder WWW bör kunna se dina sidor. Det här valet innebär också att du kan använda nästan alla koder som du kommer att lära dig i den här boken.

Om en besökare som till exempel använder Mosaic tittar på en sida som innehåller Netscapes och Internet Explorers specialkoder, blir resultatet i bästa fall att sidan ser konstig ut. I värsta fall försvinner text och bild. Det är därför viktigt att tala om för besökaren vilka webbläsare sidorna är avsedda att ses med, till exempel som i figur 2.4.

Figur 2.4
Tala om för besökaren med vilken webbläsare din hemsida ska ses.

En sida för ett fåtal

Slutligen kan du välja att göra hemsidor som måste ses med en enda webbläsare för att fungera. Boken innehåller avsnitt med specialkoder som bara Netscape 6.x respektive Internet Explorer 5.x begriper. Även Style Sheets och Dynamisk HTML (mer om dessa tekniker längre fram i boken) kräver någorlunda nya webbläsare för att fungera.

Om du väljer detta alternativ är det naturligtvis oerhört viktigt att du tydligt talar om exakt med vilken webbläsare dina sidor ska avnjutas. Du bör också fundera på om besökarna verkligen kommer att vilja ladda ned en helt ny webbläsare på uppemot 10 megabyte för att se dina sidor. Tumregeln är att ju mer intressant och välgjord din sida är, desto större krav kan du ställa på dina besökares utrustning.

Tips!
Hos Browserwatch kan du botanisera bland alla de olika webbläsare som finns. Adressen är: http://www.browserwatch.com/

Arbetsmetodik

I nästa kapitel ska du få lära dig själva kärnan i konsten att göra hemsidor: HTML-kodningen. Men för att du ska bli varm i kläderna, och dessutom förstå hur det rent tekniskt går till att göra en hemsida, ska vi göra en redan nu.

Börja med att starta både din texteditor och den webbläsare du valt att använda.

Skapa därefter ett nytt dokument i texteditorn och knappa in HTML-koden i exempel 2.5 a. För tillfället behöver du inte bry dig om vad koderna betyder, det kommer du snart att lära dig. Var noga med att skriva exakt som exemplet visar. Naturligtvis kan ingenting hända med din dator om du skulle skriva fel, men hemsidan kommer då inte att se ut som den ska. Tecknen som omger HTML-koderna är tecknen för mindre än och större än (< och >).

Exempel 2.5 a
Större än så här behöver inte en hemsida vara. Var noga med att knappa in koden exakt som i exemplet.

```
<html>
<head>
<title>Min hemsida</title>
</head>
<body>
<h1>Min hemsida</h1>
Det här är min första hemsida.<br />
Den är kort, men kärnfull.
</body>
</html>
```

När du skrivit in koden måste dokumentet sparas. Välj en plats på hårddisken som du lätt hittar tillbaka till. Enklast är att skapa en ny mapp (eller ”katalog” som det kallas i Windows) för alla de hemsidor du snart kommer att skriva.

Du måste också namnge ditt HTML-dokument. Välj ett namn som inte innehåller mellanslag, svenska tecken (å, ä och ö) eller specialtecken (till exempel ”/”, ”*” och ”$”). Namnet bör också sluta med en punkt följd av bokstäverna ”html” eller “htm”. Detta markerar att dokumentet är ett HTML-dokument som kan läsas med hjälp av en webbläsare.

När det gäller namn på hemsidor kan det göra skillnad med stora och små bokstäver. Beroende på vilket operativsystem webbservern körs på är ibland “Min_Sida.html” alltså inte detsamma som “min_sida.html”. Många HTML-kodare håller sig genomgående till små bokstäver i filnamnen, eftersom det då blir lättare att hålla reda på vad sidorna heter.

Figur 2.5 b
Hemsidan är färdig. Om de svenska tecknen byts ut mot någonting helt annat beror det troligen på att du använder en Macintosh.
I kapitel 14 kan du läsa mer om problemet, och vad du kan göra åt det.

Viktigt.
När du öppnar ett dokument med Internet Explorer för Windows måste du först välja "Open" under File-menyn och sedan klicka på knappen märkt "Browse..."

När du sparat ditt HTML-dokument är det dags att öppna det med webbläsaren. Det gör du genom att aktivera läsaren och välja ”Open” eller ”Open File” under ”File”-menyn. I väljaren bläddrar du dig sedan fram till HTML-dokumentet och öppnar det. Vad som sedan händer är i princip samma sak som i vårt exempel med hemsidan om tibetansk konst i förra kapitlet. Enda skillnaden är att HTML-dokumentet inte behöver färdas via Internet från USA till din webbläsare, utan bara från din hårddisk.

Att göra ändringar

Om allt gått som det ska bör sidan se ut ungefär som i figur 2.5 b. Låt oss nu göra en ändring i din nyskrivna sida. Gå tillbaka till texteditorn och öppna ditt HTML-dokument om det inte redan är öppet. Ändra sedan ordet ”första” till “första”. Koden ska nu se ut som i exempel 2.6 a. Spara sedan dokumentet utan att byta namn på det.

Exempel 2.6 a
Koden är precis likadan som i exempel 2.5 a, förutom att ordet "första" nu omges av de två HTML-koderna "" och "".

```
<html>
<head>
<title>Min hemsida</title>
</head>
<body>
<h1>Min hemsida</h1>
Det här är min <b>första</b> hemsida.<br />
Den är kort, men kärnfull.
</body>
</html>
```

Tips!
Du kan använda följande tangentkommandon för att snabbt fräscha upp en sida vars kod ändrats:
Internet Explorer och Netscape/PC: Ctrl + R
Netscape/Mac: Äpple + R

Att ladda in en ny version

Har du sparat dokumentet med ändringen? Gå då tillbaka till webbläsaren. För att den nya versionen av ditt dokument ska synas måste du trycka på knappen märkt ”Reload” (Netscape) eller ”Refresh” (Internet Explorer). Webbläsaren laddar nu in den nya versionen av ditt dokument.

Ser du skillnaden? HTML-koden gör stilen fet, och med koden stängs den feta stilen av igen. Att ”koppla på” en effekt med en kod och ”stänga av” effekten med en annan kod är mycket vanligt i HTML. Experimentera gärna med att flytta runt och i texten.

Det här är min **första** hemsida.
Den är kort, men kärnfull.

Figur 2.6 b
HTML-koderna och talar om för webbläsaren att ordet eller orden mellan koderna ska visas med fet stil.

Om du vill bli ännu varmare i kläderna inför nästa kapitel kan du prova att sätta koderna och före och efter ett eller flera ord och se vad som händer. Testa också att skriva in en eller flera <hr> i dokumentet. Glöm inte att spara ditt HTML-dokument så fort du gjort en ändring.

Planera din hemsida

Arbetet med att göra hemsidor består av två moment som upprepas om och om igen tills hemsidan ser ut som den ska:

- Skriv kod och spara.
- Titta på sidan med en webbläsare.
- Gå tillbaka till texteditorn och ändra i koden. Spara.
- Titta på resultatet (osv).

Det är klokt att planera hur hemsidan ska se ut innan själva kodningen börjar. Professionella kodare börjar ofta med att göra en skiss på papper över hur sidan ska se ut. Först därefter börjar man själva kodandet.

Var försiktig med att skriva mycket kod utan att kontrollera hur sidan ser ut i webbläsaren. Ett enda felplacerat tecken kan få dramatiska följder, och om du gjort stora ändringar när felet uppstår kan det ta onödigt lång tid att hitta misstaget. Av denna orsak brukar även rutinerade HTML-kodare kontrollera sina sidor med mycket täta intervall.

Låt dig inte luras av cache-minnet!

Om du använder Netscape händer det ibland att en hemsida inte förändras när du klickar på ”Reload” trots att du vet att du ändrat i HTML-koden. Då är det webbläsarens cache-minne som spökar. Gör så här:

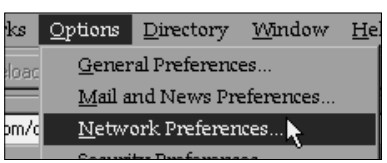
Öppna ”Network Preferences” under alternativet ”Options” i Netscapes meny (figur 2.7).

Klicka på ”Cache” (figur 2.8).

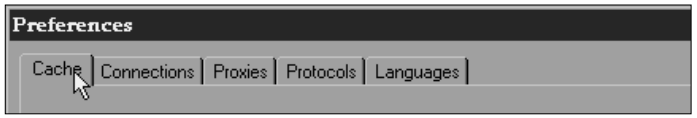
Här kan du ställa in hur ofta Netscape ska kontrollera om en hemsida har ändrats. Se till att ”Every Time” är förkryssat vid rubriken ”Verify Documents” (figur 2.9).

Nu kommer Netscape att kontrollera dokumenten varje gång du trycker på ”Reload”.

Cache. För att inte Netscape ska behöva ladda ned samma information om och om igen när du återbesöker hemsidor sparas alla sidor du besöker i programmets cache-minne. Detta är oftast en fördel, men om en hemsida ändras ofta riskerar du att inte få se senaste versionen.



Figur 2.7
Öppna ”Network Preferences” under ”Options” i Netscapes meny.



Figur 2.8
Klicka på ”Cache”.

Figur 2.9
Klicka på ”Every Time”. Nu kommer Netscape att kontrollera om HTML-koden i dina hemsidor ändrats varje gång du laddar ett dokument.

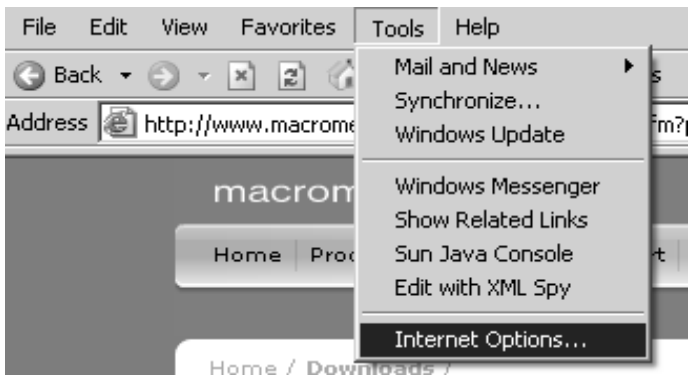
För motsvarande procedur för Internet Explorer, gör så här:

Öppna “Internet options” under fliken “Tools” i Internet Explorers meny.

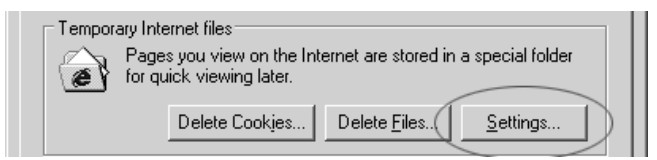
Klicka på “Settings”.

Välj “Every visit to the page” under alternativet “Check for newer versions of stored pages”.

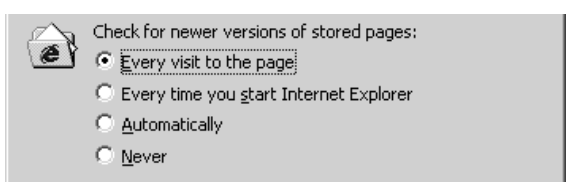
Nu letar Internet Explorer upp den senaste versionen av sidan varje gång du laddar den.



Figur 2.10
Öppna ”Internet Options” under ”Tools” i Internet Explorers meny.



Figur 2.11
Välj ”Settings” under alternativet “Temporary Internet files”.



Figur 2.12
Välj ”Every visit to the page” under alternativet “Check for newer versions...”.

Stjäl html-kod!

När du är ute och tittar på andras hemsidor på WWW kan du alltid se hur en hemsida du gillar är kodad. Välj bara ”View Document Source” under alternativet ”View” i webbläsarens meny och hela koden blir synlig. Sedan är det bara att kopiera de HTML-avsnitt du tycker är intressanta. Om du vill spara koden till en hel hemsida gör du så här: Välj ”Save As…” eller ”Save As File...” (beroende på vilken webbläsare du använder) under menyalternativet ”File”. När dialogrutan kommer upp väljer du ”HTML” som filformat och sparar dokumentet på hårddisken. Sedan är det bara att öppna dokumentet i en texteditor och plocka de HTML-godbitar du hittar.

3 DIN FÖRSTA HEMSIDA: Grunderna

Nu är det dags att börja skriva HTML på allvar. I det här kapitlet kommer vi att gå igenom allt du behöver veta för att göra en hemsida med rubriker i olika storlekar och text som är formaterad på olika sätt. Men vi börjar med att titta på hur ett HTML-dokument är uppbyggt.

HTML, Hyper Text Markup Language, är ett sidbeskrivningsspråk. Precis som vilket språk som helst innehåller det särskilda ord, eller ”HTML-koder” som vi kommer att kalla dem i den här boken. Rätt använda talar dessa koder om för en webläsare hur din hemsida ska se ut. Det finns särskilda koder för varje moment på hemsidan, till exempel för var bilderna ska placeras och hur texten ska se ut.

I förra kapitlet tittade vi på HTML-koderna och . All text som placeras mellan dessa båda koder visas med fet stil (*bold* på engelska, därav bokstaven ”b”). Exempel 3.1 a och figur 3.1 b visar hur används och hur resultatet blir i en webläsare.

Exempel 3.1 a
Koden gör texten fet...

```
Det här är min <b>första</b> hemsida.
```

Figur 3.1 b
...och här är resultatet.

 är en mycket typisk HTML-kod. Som alla andra koder omges den av ”<” och ”>”. Och precis som i flertalet HTML-koder har den en avslutningskod som talar om för webläsaren att texten inte ska vara fet längre.

Avslutningskoden är identisk med startkoden men inleds med ett snedstreck: . Ett sätt att se på HTML-kodning är att den första koden, i det här fallet , ”slår på” eller ”startar” en särskild funktion, i det här fallet fet text. Avslutningskoden ”stänger av” funktionen.

Hemsidans beståndsdelar

Svenska tecken. Om de svenska tecknen (å, ä och ö) ser konstiga ut på din hemsida kan det bero på att du använder Macintosh. Bläddra i så fall fram till kapitel 18, där lösningen på problemet finns.

Ett HTML-dokument har två typer av innehåll: koderna och texten. Koderna är alltid osynliga när webläsaren visar hemsidan. Allting som inte omges av ”<” och ”>” betraktar webläsaren däremot som text, och den visas på hemsidan. HTML-koderna kan ha många olika funktioner. Vissa fungerar som och bestämmer hur den synliga texten ska se ut, medan andra talar om för webläsaren att en bild ska visas på hemsidan och var på Internet denna bild finns.

En tredje typ av koder påverkar överhuvudtaget inte hemsidans utseende men är nödvändig att ha med ändå. Dessa koder är till för att organisera HTML-dokumentet på ett sätt som alla webläsare förstår.

För att se hur en hemsida är uppbyggd ska vi titta närmare på HTML-dokumentet i exempel 3.2 a. Öppna ett nytt textdokument och knappa in koden precis som den ser ut i exemplet. Spara dokumentet på hårddisken och öppna det sedan med din webläsare. Sidan bör se ut ungefär som i figur 3.2 b.

För att bekanta oss med hemsidans uppbyggnad ska vi på nästa sida gå igenom koden i exempel 3.2 a rad för rad.

Exempel 3.2 a
Knappa in koden exakt som den står. Var noga med att inte av misstag skriva två ”<” eller ”>” i rad, det kan ge märkliga resultat. De små siffrorna till vänster är radnummer och ingår inte i koden.

```
1.  <html>
2.  <head>
3.  <title>Min hemsida</title>
4.  </head>
5.  <body>
6.  <h1>Min hemsida</h1>
7.  Det här är min <b>första</b> hemsida.<br />
8.  Den är kort, men kärnfull.<p>
9.  Detta är ett nytt stycke.
10. </body>
11. </html>
```


Figur 3.2 b
En komplett hemsida! De allra mest avancerade hemsidorna görs efter exakt samma princip som den här enkla övningssidan.
Lägg märke till att hemsidans titel syns i den smala listen överst i fönstret.

Sökmotor. En sökmotor är ett program som automatiskt surfar runt och katalogiserar innehållet på hemsidor runt om i världen. Sedan kan vem som helst använda registret för att med hjälp av sökord hitta sidor med särskild information. Så fort du publicerat din sida på World Wide Web kommer den alltså att få besök av flera sökmotorer. Ofta registreras all text som finns på sidan, men ibland är det bara texten i titeln som hamnar i registret. Du hittar adressen till flera bra sökmotorer på bokens hemsida.

Rad 1: <html>

Webläsarna läser HTML-dokument precis som vi; de börjar uppifrån och läser från vänster till höger. Det första webläsaren stöter på i vårt exempel är således koden <html>. Denna kod inleder alla HTML-dokument och talar om för webläsaren att den nu ska förbereda sig på att visa en hemsida. Om man säger att den vid det här laget välbekanta koden ”startar” funktionen fet stil kan <html> alltså sägas starta hela hemsidan. När webläsaren nått sista raden i dokumentet stängs hemsidan av med avslutningskoden </html>.

Rad 2 till 4: <head> och <title>

En hemsida består av två delar: huvudet och kroppen (*head* och *body* på engelska). Huvudet kommer logiskt nog först och startas med koden <head>. Huvudet innehåller i 99 fall av 100 bara en enda sak: hemsidans titel. Titeln är en valfri text som står mellan koderna <title> och </title>. Titeln placeras automatiskt i webläsarens övre list när hemsidan visas. Texten bör väljas med omsorg eftersom många **sökmotorer** fäster stor vikt vid ett dokuments titel. Om du döper ditt dokument till något som faktiskt har med innehållet att göra ökar alltså chansen att andra människor lyckas hitta fram till din sida. Om du gör en privat hemsida kan det vara idé att ha med ditt namn i titeln.

När titeln är avklarad finns det inget mer att göra i hemsidans huvud, som därför avslutas med avslutningskoden </head>.

Rad 5: <body>

När hemsidans huvud är avslutat kommer kroppen, som inleds med koden <body>. Kroppen är den viktigaste och största avdelningen i ett HTML-dokument. Här återfinns allt det som faktiskt kommer att synas på själva hemsidan: text, rubriker och bilder. Det innebär att nästan alla HTML-

koder du kommer att lära dig i den här boken hör hemma i HTML-dokumentens <body>-avdelningar.

Rad 6: <h1>

Sidans egentliga innehåll inleds på rad sex. Texten ”Min hemsida” är en rubrik, vilket framgår av att den omges av två nya koder: rubrikkoderna <h1> och </h1>. Bokstaven H är en förkortning av engelskans ord för rubrik, *heading*. Om du byter ut siffran 1 mot 2, 3, 4, 5 eller 6 kommer rubriken att minska i storlek. Vi återkommer strax till hur rubrikkoden används.

Rad 7: och

Svenska tecken.
Om dina svenska tecken (å, ä och ö) ser konstiga ut på hemsidan kan det bero på att du använder Macintosh. Bläddra i så fall fram till kapitel 18, där lösningen på problemet finns.

Efter rubriken följer vanlig text. HTML-koderna och har vi redan talat om, de gör så att ett av orden sätts med fet stil. Sist i samma rad kommer en av de koder i HTML-språket som avslutas lite annorlunda: <br />. Den utför en radbrytning (*break* på engelska). Prova att sätta in fler <br /> på olika ställen i texten och se vad som händer.

Rad 8-9:

På åttonde raden hittar du HTML-koden för nytt stycke, (*paragraph* på engelska). -koden gör ungefär samma sak som <br /> förutom att radmellanrummet blir lite större. har en avslutningskod, .

Rad 10 och 11: Avslutning

Sist i vårt exempel avslutas först HTML-dokumentets kropp med avslutningskoden </body> och därefter hela HTML-dokumentet med </html>.

Gör ett malldokument

Vissa HTML-koder ingår som vi tidigare sagt i samtliga HTML-dokument. Detta gäller koderna <html>, <head>, <title>, och <body> med respektive avslutningskoder. Du kan spara en hel del tid genom att skapa ett malldokument där alla dessa koder ingår. På så sätt behöver du inte knappa in standardkoderna varje gång du ska göra en ny hemsida.

Malldokumentet kan se ut som i exempel 3.3. Knappa in koden och spara dokumentet på hårddisken. Du kan exempelvis döpa det till ”sidmall.htm”.

Exempel 3.3 Malldokumentet.
Att slippa skriva in standardkoderna i varje HTML-dokument sparar mycket tid. Med hjälp av ett malldokument slipper du rutinarbetet.

```
<html>
<head>
<title>Här ska dokumentets titel stå</title>
</head>
<body>

Här ska hela hemsidans innehåll in.

</body>
</html>
```

För att inte riskera att av misstag skriva över malldokumentet kan det vara bra att skrivskydda det. Gör så här:

Macintosh

Markera dokumentet genom att klicka en gång på det. Tryck sedan samtidigt på tangenterna ”command” och ”i”. Command-tangenten är markerad med ett äpple och finns till vänster och höger om mellanslagstangenten. Om du tryckt rätt kommer du att se ett litet fönster som innehåller information om det markerade dokumentet. Kryssa i rutan märkt ”Malldokument” i fönstrets nedre högra hörn. Mallen är nu skrivskyddad.

Windows

Leta upp dokumentet som ska skrivskyddas och klicka på det med högerknappen. Välj ”Egenskaper” i den mindre menyn som nu visas. Kryssa i rutan märkt ”Skrivskyddad” i informationsfönstret.

Några ord om bokens kodexempel

Viktigt.
På bokens hemsida hittar du samtliga kodexempel som finns med i boken. Dessutom finns där information och kompletteringar som ytterligare kan underlätta ditt HTML-skrivande. Du når sidan på adress: http://www.etc.se/nyckeln/

I fortsättningen kommer standardkoderna av utrymmesskäl inte att finnas med i bokens kodexempel. Istället utgår vi ifrån att du använder ett malldokument och skriver in kodexemplen i dokumentets <body>-del.

Se bokens kodexempel som vinkar i en viss riktning snarare än exakta instruktioner som du slaviskt måste följa. Får du lust att testa någon egen variant eller kanske kombinera koder på ett sätt som inte tas upp bland exemplen – gör det! Det allra bästa sättet att lära sig HTML är att experimentera friskt, och det finns ingen som helst risk att datorn eller programmen går sönder hur vilda dina experiment än blir. Det värsta som kan hända är att din hemsida ser annorlunda ut än du tänkt dig, och det kan ibland innebära positiva överraskningar. Glöm inte heller att World Wide Web vimlar av exempel på hur HTML-koder kan användas på innovativa sätt. Att surfa är med andra ord mycket lärorikt.

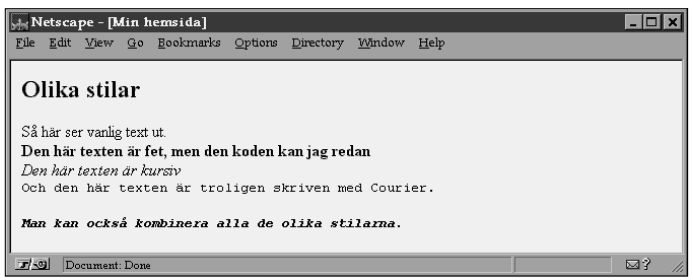
Att formatera text i HTML

Nu är det dags för en komplett genomgång av de HTML-koder som används för att på olika sätt formatera text på din hemsida. Vi börjar med tre olika stilsorter som man ofta använder. Öppna malldokumentet och knappa in koden i exempel 3.4 a i HTML-dokumentets <body>-del. Spara dokumentet och öppna det med din webläsare. Resultatet bör likna figur 3.4 b.

Exempel 3.4 a
Det finns flera olika sätt att ändra utseendet på en text.

```
<h2>Olika stilar</h2>
<b>Den här texten är fet, men den koden kan jag redan.</b>
<br />
<i>Den här texten är kursiv.</i><br />
<tt>Och den här texten är skriven med Courier.</tt><p>

<b><i><tt>Man kan också kombinera alla de olika
stilarna.</tt></i></b>
```

Figur 3.4 b
Resultatet.



Som du ser gör texten kursiv på samma sätt som skapar fet stil. Bokstaven I är en förkortning av engelskans ord för kursiv, *italic*. Koden <tt> är lite mer komplex. Den säger till webläsaren att texten ska visas med ett typsnitt som har fast bredd. I vanliga fall är typsnitt på hemsidor, liksom i den här boken, proportionella. Det innebär att bokstäverna tar upp olika mycket plats beroende på hur de är uppbyggda. ”O” och ”A” är till exempel betydligt bredare än ”I”. Typsnitt med fast bredd ger precis som en skrivmaskin alla bokstäver lika mycket utrymme.

Det typsnitt nästan alla webläsare använder för att visa text med fast bredd heter Courier och liknar skrivmaskinstext.

Anledningen till att HTML-koden <tt> finns är att HTML utvecklades av forskare och datortekniker som behövde visa datakod på hemsidorna. Sådan text visas traditionellt med fast bredd för att den lätt ska kunna skiljas från annan text. Nuförtiden finns det många hemsidesmakare som tycker att den lite kantiga skrivmaskinstexten är snygg och därför använder den till helt andra saker än datakod.

Som framgår av sista raden i exempel 3.4 a finns det inget som hindrar att flera olika stilsorter kombineras med varandra. Experimentera gärna med olika kombinationer, men glöm inte att slutkoderna , och </tt> alltid måste vara med.

Om du använder flera formateringskoder på en gång kan det också vara klokt att skriva avslutningskoderna i rätt ordning. Titta en gång till på sista raden i exempel 3.4 a:

```
<b><i><tt>Man kan också kombinera alla de olika
stilarna.</tt></i></b>
```

Eftersom står ytterst till vänster på raden ska den också stå ytterst till höger när textformateringen ska ”stängas av” med hjälp av avslutningskoderna. På samma sätt behåller och <tt> sina positioner även som avslutningskoder. Om man ändrar ordningen och till exempel skriver

```
<b><i><tt>Man kan också kombinera alla de olika
stilarna.</b></tt></i>
```

kan vissa webläsare bli förvirrade och helt hoppa över en eller flera formateringskoder. För att undvika detta bör man alltid tillämpa principen först in – sist ut. I xhtml är detta obligatoriskt, läs mer om detta i kapitel 19.

Ovanligare textformateringar

Det finns ytterligare några textformateringar som inte används så ofta men som ändå kan vara bra att känna till. Prova själv att kombinera dem med varandra.

Kod	Funktion
<s>…</s>	Skapar genomstruken text (*strikethrough* på engelska).
<u>…</u>	Skapar understruken text. Kan lätt förväxlas med klickbara länkar och bör därför användas sparsamt.
_…	Skapar en mindre, nedsänkt text (*subscript* på engelska).
[…]	Skapar en mindre, upphöjd text (*superscript* på engelska).
<blink>…</blink>	Får texten att blinka. (endast Netscape)

Tips!
Du får en fingervisning om vad folket på Netscape själva tycker om <blink> genom att skriva "about:mozilla" (utan citationstecken) i Netscapes adressfönster och sedan trycka på return.

Koden som får texten att blinka – <blink> – är antagligen den mest hatade koden i hela HTML-språket, och den skapades mest som ett skämt av Netscape. Detta illustreras av att koden länge saknades i företagets egen lista över gångbara HTML-koder, och att fortfarande ingen annan webläsare stödjer blinkande text. Trots detta spred sig <blink>-användandet till mångas förtvivlan snabbt över World Wide Web, och idag finns det till och med hemsidor som i protest pryds av symboler med texten "garanterat blinkfri zon". Anledningen till det starka missnöjet är att det kan vara irriterande att titta på sidor med blinkande text. Använd <blink>-koden med största försiktighet och låt aldrig mer än ett eller högst ett par ord blinka.

Olika sätt att dela upp texten

När du nu vet hur det går till att få texten på en sida att ändra utseende är det dags att titta på de olika sätt som finns att dela upp texten så att den blir lättläst. Det finns fyra olika metoder att göra detta:

- Med en rubrik
- Med en radbrytning
- Med ett nytt stycke
- Med en horisontell linje

Figur 3.5 a och b visar en sida där de flesta av dessa metoder används.

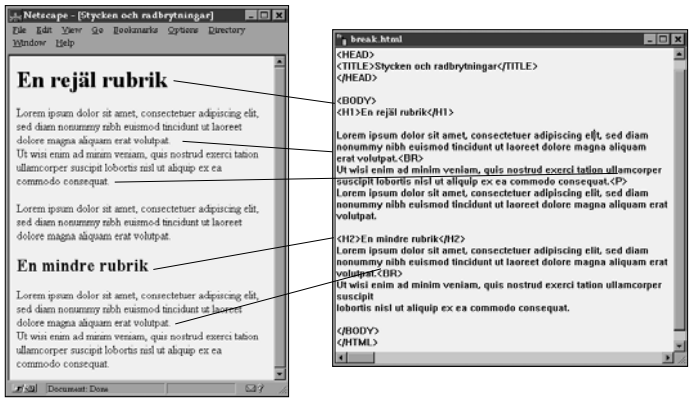
Rubriker

Rubriker är text som markerats med rubrikkoden <h*värde*> där värdet är en siffra mellan ett och sex. Ju högre siffran är, desto mindre blir rubriken. <h1> är alltså största och <h6> minsta rubrikvarianten. Rubrikens avslutningskod måste innehålla samma värde som startkoden. En rubrik som startas med <h2> måste alltså avslutas med </h2> och inget annat.

Titta på figur 3.5 a och b. Sidan innehåller två rubriker: "En rejäl rubrik" som har storlek 1 och "En mindre rubrik" som håller storlek 2 Gemensamt för båda är att de har sin egen rad och ett rejält utrymme till både föregående och efterföljande text. Rubriker har inbyggd radbrytning, och avståndet till texten under beror på rubrikens storlek.

Öppna gärna ett nytt HTML-dokument och experimentera med olika rubrikstorlekar. Du kommer att märka att de minsta rubrikerna faktiskt är mindre än normalstor text. Mindre rubriker visas dock med fet stil för att skilja sig från övrig text.

Figur 3.5 a (höger)
Det finns olika sätt att dela upp en text i sektioner. Här visas rubriker, radbrytningar och nytt stycke.
Figur 3.5 b (vänster)
Resultatet av koden i figur 3.5 a.



Radbrytning och nytt stycke

När en webläsare stöter på en lång text i ett HTML-dokument bryter den själv raderna där det behövs för att texten ska få plats i fönstret. När du däremot vill att en rad ska brytas på ett visst ställe krävs det en särskild kod, som redan nämnts:
 (från engelskans *break*). Som framgår av figur 3.5 a och b ger <br /> inget större radmellanrum, bara ett radbyte.

När det är dags att byta stycke, vilket brukar markeras med ett lite större radmellanrum, används istället koden <p> (efter engelskans *paragraph*). Effekten bli densamma som med <br />, med den skillnaden att texten som följer efter <p> hamnar en bit längre ned.

<p> låter dig dessutom bestämma om den efterföljande texten ska vara centrerad eller högerjusterad. <p align="center"> (justera heter *align* på engelska) centrerar texten och <p align="right"> högerställer den.

Attribut till HTML-koder

Att som i exemplet <p align="right"> lägga till en extrakod inuti en annan kod kallas att använda ett *attribut*. align="center" är alltså ett attribut till koden <p>. Du kommer snart att märka att många HTML-koder har attribut med vilka man mer exakt kan specificera vad huvudkoden ska åstadkomma. Attributet kan styra allt från storlek på text till en hemsidas bakgrundsfärg, bildramars tjocklek och mycket annat. Attributet skrivs nästan alltid i formen <*kod attribut="värde"*>.

Horisontella linjer

Det sista sättet att dela upp texten på en hemsida är både enkelt och kraftfullt: lägg in en horisontell linje. HTML-koden som åstadkommer detta heter <hr /> (efter engelskans *horizontal rule*), och skapar en linje som sträcker sig över hela sidan och är en pixel hög. Någon avslutningskod finns inte, och precis som rubriker hamnar en linje alltid automatiskt på en egen rad. Du behöver alltså inte använda några koder för radbrytning eller nytt stycke vare sig framför eller efter en horisontell linje.

Om inte standardlinjen passar din hemsida finns det rika möjligheter att förändra linjens utseende med dessa attribut:

size=*värde* talar om hur tjock linjen ska vara mätt i pixlar.

width=*värde* eller *värde%* talar om hur bred linjen ska vara, antingen uttryckt i antal pixlar eller i procent av webläsarens fönsterbredd.

align=*left* eller *right* bestämmer om linjen ska vara vänsterställd eller högerställd.

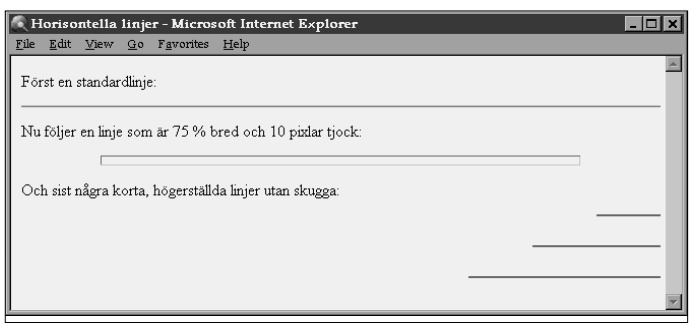
noshade tar slutligen bort standardlinjens skugga, som ger ett 3D-liknande utseende.

Exempel 3.6 a och figur 3.6 b visar hur <hr /> och alla kodens attribut används. Lägg märke till att en linje vars bredd (width) angivits som procent får olika längd beroende på hur stort webläsarens fönster är.

Exempel 3.6 a
Horisontella linjer.
Det enklaste sättet att dela upp en sida i tydligt åtskilda avdelningar är med horisontella linjer. Koden <hr /> har många attribut som påverkar linjens utseende.

```
Först en standardlinje:
<hr />
Nu följer en linje som är 75 % bred och 10 pixlar tjock:
<hr size="10" width="75%" />
Och sist några korta, högerställda linjer utan skugga:
<hr size="2" width="10%" align="right" noshade />
<hr size="2" width="20%" align="right" noshade />
<hr size="2" width="30%" align="right" noshade />
```

Figur 3.6 b
Resultatet av kodningen i exempel 3.6 a.



Förformatera texten med <pre>

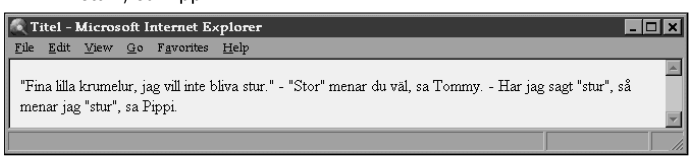
För dig som har ont om tid erbjuder koden <pre> möjlighet att slippa använda koderna för radbrytning och nytt stycke. Namnet är en förkortning av *preformated*, förformaterad, och koden gör så att texten återges så som den ser ut i själva HTML-dokumentet – komplett med radbrytningar. <pre> upphäver med andra ord regeln att radbrytningar på en hemsida måste åstadkommas med koderna <br /> och <p>. Nackdelen är att <pre> inte kan kombineras med några andra formateringar, och att du därför inte kan lägga in rubriker eller liknande. Mest lockande är <pre> om du ska publicera en lång text på hemsidan och inte orkar knappa in alla <br /> och <p> som normalt sett behövs.

Vad är det då <pre> gör? Exempel 3.7 a och figur 3.7 b visar hur webläsaren behandlar en vanlig text som inte innehåller några koder för radbrytningar eller nya stycken.

Exempel 3.7 a
En text utan koder för radbrytningar och nya stycken...

```
"Fina lilla krumelur,
jag vill inte bliva stur."

-"Stor" menar du väl, sa Tommy.
- Har jag sagt "stur", så menar jag
"stur", sa Pippi.
```

Figur 3.7 b
... hamnar på en rad när webläsaren visar hemsidan.



Nu lägger vi till <pre> före citatet ur Pippi Långstrump i Söderhavet, och avslutningskoden </pre> (efter exempel 3.8 a). Figur 3.8 b visar den dramatiska skillnaden.

Exempel 3.8 a
När <pre> används tar webläsaren hänsyn till radbrytningarna i själva HTML-koden.

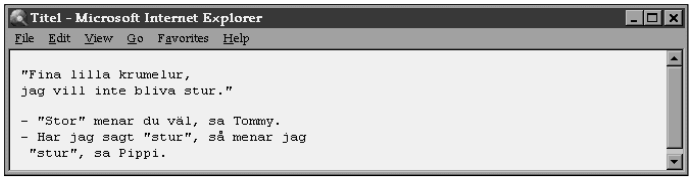
```
<pre>"Fina lilla krumelur,
jag vill inte bliva stur."

- "Stor" menar du väl, sa Tommy.
- Har jag sagt "stur", så menar jag
"stur", sa Pippi.</pre>
```

<pre> får webläsaren att ta hänsyn till de radbrytningar som finns i själva HTML-dokumentet. <pre> har dock klara begränsningar. Som du ser i figur 3.8 b visas texten med Courier, precis som om du formaterat den med HTML-koden <tt>. Du kan inte heller använda några HTML-

koder förutom hyperlänkar (mer om dem i ett senare kapitel) i text som formateras med <pre>.

<pre> kan kompletteras med attributet WIDTH=”*värde”* där värdet är det maximala antalet tecken varje rad inom <pre>-markeringen får innehålla.

Figur 3.8 b
<pre> får webläsaren att ta hänsyn till de radbrytningar som finns i själva HTML-dokumentet. Resultatet blir ett enkelt och snabbt sätt att få längre texter läsbara från hemsidan utan att behöva använda koderna för styckeindelning.



Att ändra storlek på text

Hittills har vi bara ändrat storlek på rubrikerna, men det finns naturligtvis också möjlighet att styra storleken på vanlig text. Du kan välja mellan sju olika textstorlekar. Om ingen storlek anges (som hittills i kodexemplen) visas texten automatiskt med storlek 3.

För att styra textstorleken ska vi introducera en helt ny HTML-kod: . Det är också den första koden vi hittills använt som kräver ett attribut för att göra någonting överhuvudtaget. Attributet som används för att ändra textstorlek är size=*värde*. Värdet kan vara från 1 (minst) till 7 (störst). Lägg märke till att detta är tvärtemot rubrikstorlekarna, där värdet 1 ju gav den största rubriken och 6 den minsta.

Exempel 3.9 a och figur 3.9 b visar hur används. Lägg märke till att textstorleken alltid återgår till normalstorleken 3 så fort du använder slutkoden .

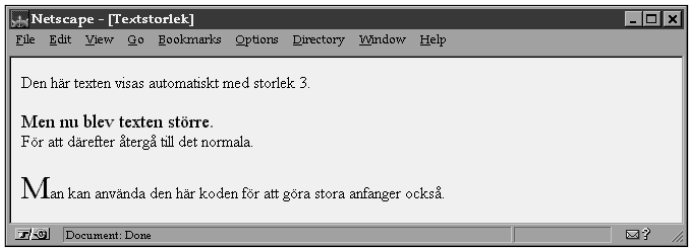
Exempel 3.9 a
<font size=*”värde”*> – konsten att ändra storlek på text. Glöm inte att storlek 1 är minst och 7 störst. För rubriker är det tvärtom. Resultatet av kodningen hittar du i figur 3.9 b.

```
Den här texten visas automatiskt med storlek 3.<p>
<font size="4">Men nu blev texten större.</font><br />
För att därefter återgå till det normala.<p>

<font size="6">M</font>an kan använda den här koden för att
göra stora anfanger också.
```

Experimentera gärna med att kombinera olika storlekar på text med koderna för textformatering som vi gick igenom i början av kapitlet. Du

kommer snart att märka att rubriker lika gärna kan skapas med hjälp av <font size=”*värde”*> som med rubrikkoden <h*värde*>. Enda skillnaden är att du måste göra en egen radbrytning om du använder <font size=”*värde”*>. Detta sker ju bara automatiskt om du använder den riktiga rubrikkoden <h>.

Figur 3.9 b
Resultatet av kodningen i exempel 3.9 a.



Viktigt.
Du kan aldrig veta hur stor textstorleken 3 faktiskt blir på skärmen hos dem som besöker din hemsida. Webläsaren kan nämligen ställas in så att ”storlek 3” blir precis så stor användaren önskar. Även om <font size=värde> ger en viss kontroll måste du alltså se till att din hemsida ser någorlunda bra ut oavsett textstorlek. Ett bra sätt att kontrollera detta är att på försök ändra inställningen i din egen webläsare så att texten visas större eller mindre än normalt.

Att ändra textens standardstorlek

Utgångsläget är som tidigare nämnts att en text visas med storlek 3, om inte storleken ändrats med koden <font size=”*värde”*>. Du kan dock ändra på standardstorleken. Koden heter <basefont> och använder liksom <font> attributet size=”*värde”*. <basefont> saknar avslutningskod. Värdet du anger blir det nya standardvärdet för all efterföljande text, och kan vara mellan 1 och 7. Basefont är mest användbar när du vill att en hel sida genomgående ska innehålla större eller mindre text. Genom att till exempel ange <basefont size=”4”> i början av HTML-dokumentet blir texten ett snäpp större utan att du behöver använda <font size=”*värde”*> en enda gång.

Stegvis ändring av textstorlek

Med hjälp av <font size=”*värde”*> kan du även ändra teckenstorleken stegvis. Lägg bara till ett plus eller minus framför värdet, och teckenstorleken ändras det antal steg du angivit i förhållande till den basefont-storlek som för tillfället gäller. <font size=”+1”> betyder till exempel att den efterföljande texten ska vara ett steg större än basefont-värdet. Om du inte angivit någon <basefont> är utgångsläget som vanligt teckenstorlek 3. Om du istället vill minska teckenstorleken skriver du så här: <font size=”–1”>.

Det är viktigt att komma ihåg att den stegvisa ökningen eller minskningen av textstorlek alltid sker i förhållande till basefont-storleken, inte i förhållande till den föregående texten. Om du till exempel skriver <font size=”+1”> två gånger i rad kommer texten inte att öka två steg i storlek,

utan ett. Om du tycker att detta är svårt att förstå så titta noga på koden i exempel 3.10 a och resultatet i figur 3.10 b.

Exempel 3.10 a
<font size="+1"> ökar textstorleken med ett steg i förhållande till rådande basefont-storlek. Utgångsläget är 3 om ingen basefont angivits.

```
<basefont size="4">
Med <font size="+1">den <font size="+2">här <font
size="+3">koden
<font size="+2">kan <font size="+1"> du <font size="4">ändra
<font size="–1"> textstorleken
<font size="–2"> i <font size="–3">steg.<p>
```

Figur 3.10 b
Resultatet av kodningen i exempel 3.10 a.

Att bestämma typsnitt

Med kan du själv bestämma typsnitt för ett textavsnitt, ord eller till och med en enskild bokstav. Texten återgår till standardtypsnittet efter avslutningskoden . För att en text ska visas med fet och kursiv Verdana skrivs koden så här:

Exempel 13.11 a
Med attributet face="*typsnitt*" ändrar du typsnitt.

```
Man kan numera <font face="verdana"><b><i>ändra
typsnitt</i></b></font> på hemsidan.
```

Figur 13.11 b
Resultatet av kodningen i exempel 13.11 a.

Viktigt.
Eftersom attributet FACE i exemplet till höger har fler än ett värde (en uppräkning av tre olika typsnitt) *måste värdena stå inom parentes.*

För att face="*typsnitt*" ska fungera måste den som tittar på din sida ha det angivna typsnittet i sin dator. För att öka sannolikheten kan du räkna upp flera alternativa typsnitt, åtskilda av kommatecken:

```
<font face="typsnitt 1, typsnitt 2, typsnitt 3">
```

Om inte typsnitt 1 finns i datorn försöker webläsaren hitta typsnitt 2, därefter typsnitt 3 och så vidare. Du kan ha med hur många alternativ som helst i listan. Om inget av typsnitten finns använder webläsaren sitt standardtypsnitt.

En bra typsnittskombination för rubriker är:

```
<font face="arial, helvetica">
```

Tips!
Typsnittspaketet finns för gratis nedladdning för både PC och Mac på adress http://www.microsoft.com/msdownload/

Om typsnittet Arial finns installerat visas rubriken med Arial. Om inte Arial finns installerat men Helvetica finns, visas Helvetica.

Microsoft har släppt ett gratis typsnittspaket (”TrueType Fonts on the Web”) med identiska typsnitt till både PC och Mac. Figur 3.12 visar vad som ingår i paketet. Ett sätt att få kontroll över din hemsidas utseende är att hålla dig till dessa typsnitt och uppmana alla besökare att också ladda ned typsnittspaketet. Alla typsnitten ingår dessutom i Windows .

Figur 3.12
Typsnittspaketet innehåller sju typsnitt, varav fem är särskilt lämpade för rubriker.

Verdana, **Verdana fet**, *Verdana kursiv*
Verdana fet kursiv
Arial, **Arial fet**, *Arial kursiv*, ***Arial fet kursiv***
Arial Black, *Arial Black kursiv*
Times New Roman, **Times New Roman Fet**, *Times New Roman kursiv*, ***Times New Roman fet kursiv***

Courier New, **Courier New fet**, *Courier New kursiv*, ***Courier New fet kursiv***
Helvetica, **Helvetica fet**, *Helvetica kursiv*, ***Helvetica fet kursiv***

Att justera texten på hemsidan

Genom att använda rubriker, koderna för radbrytning och nytt stycke och horisontella linjer delas hemsidan upp i olika sektioner i höjdled. Men ibland kan det vara användbart att också markera olika avsnitt genom att förskjuta texten i sidled. Detta kallas att *justera* texten och innebär oftast att texten högerställs, vänsterställs eller centreras. All text är naturligtvis som standard vänsterställd.

Ett sätt att justera text har vi redan talat om: genom att använda attributet align="*right*" eller *"center"* tillsammans med koden för nytt stycke, <p>. Detta får den efterföljande texten att bli högerställd eller centrerad. Problemet med denna metod är att man alltid får ett nytt stycke med det tillhörande stora radmellanrummet på köpet. De tre koderna , och <blockquote> som vi nu ska titta på har ingen sådan bieffekt.

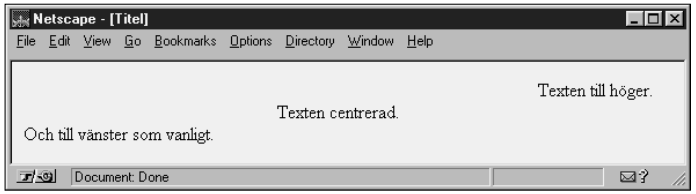
Justera text och bild med

Koden … använder sig av attributet align=*left*, *center* eller *right*. Exempel 3.13 a visar hur används och figur 3.13 b resultatet.

Exempel 3.13 a
Men koden div kan texten centreras eller justeras till vänster eller höger.

```
<div align="right">Texten till höger.</div>
<div align="center">Texten centrerad.</div>
<div align="left">Och till vänster som vanligt.</div>
```

Figur 3.13 b
Resultatet av kodningen i exempel 3.13 a.



Centrera med

Om du bara är intresserad av att centrera text och bild är enklaste metoden att använda koden … :

```
<center>
Texten är centrerad.
</center>
```

Precis som med påverkas både efterföljande text och bild, vilket gör till en mycket enkel och kraftull kod att använda. Ett enda i början på HTML-dokumentet centrerar hela hemsidans innehåll.

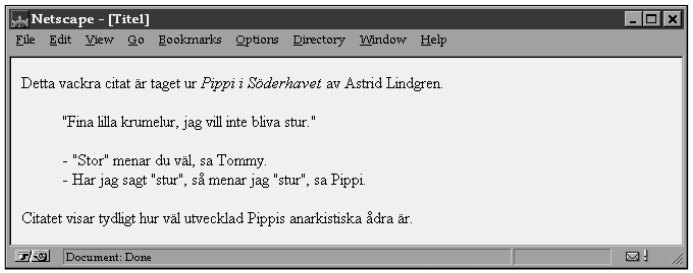
Visa citat med <blockquote>

Ibland kan det vara effektivt att markera delar av en text med hjälp av bredare vänster- och högermarginaler. Koden som åstadkommer detta heter <blockquote> och är särskilt lämpad för citat. Exempel 3.14 a och figur 3.14 b visar hur <blockquote> påverkar texten.

Exempel 3.14 a
Citat kan särskiljas från övrig text med hjälp av <blockquote>.

```
Detta vackra citat är taget ur <i>Pippi i Söderhavet</i> av Astrid
Lindgren.<p>
<blockquote>
”Fina lilla krumelur,
jag vill inte bliva stur.”<p>
- ”Stor” menar du väl, sa Tommy.<br />
- Har jag sagt ”stur”, så menar jag
”stur”, sa Pippi.</p>
</blockquote>
Citatet visar tydligt hur väl utvecklad Pippis anarkistiska ådra är.
```


Figur 3.14 b
Resultatet av <blockquote> är att vänster och höger marginal ökar i bredd, vilket gör texten indragen.



Två användbara specialkoder

Vi avslutar det här kapitlet med två specialkoder som du kanske inte kommer att ha användning för så ofta, men som är mycket användbara när de väl behövs. Den första heter <nobr> och hindrar webläsaren från att göra radbrytningar. Den andra heter <!> och ger dig möjlighet att lägga till kommentarer i dina HTML-dokument.

Hindra radbrytningar med <nobr>

Som tidigare sagts tar webläsaren normalt själv hand om radbrytningarna så att långa texter får plats i fönstret. Ibland kan det dock hända att man inte vill låta webläsaren bryta vissa ord- eller sifferkombinationer. Det blir till exempel inte snyggt om talet ”150 000” skulle brytas efter första nollan så att ”150” kommer på en rad och nästa inleds med ”000”.

<nobr>, som är en förkortning av engelskans *no break*, förhindrar detta. All text som placeras mellan <nobr> och avslutningskoden </nobr> hamnar på en enda rad, utan radbrytningar. En lösning på sifferproblemet ovan är alltså att skriva så här: <nobr>150 000</nobr>. Nu kommer hela talet att hoppa ned en rad istället för att brytas mitt i.

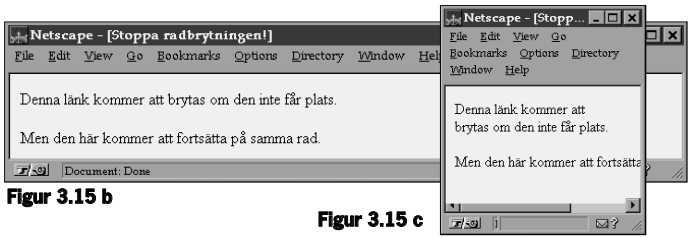
Tänk på att en lång mening kanske inte får plats i webläsarens fönster om du hindrar radbrytningar. Du kan inte heller vara säker på exakt hur lång en mening blir på skärmen, eftersom detta ju beror på hur webläsaren är inställd.

Figur 3.15 a, b och c (på nästa sida) visar hur <nobr> fungerar. Lägg märke till att raden utan radbrytningar inte får plats i fönstret när detta görs mindre.

Exempel 3.15 a
Med <nobr> hindras webläsaren att göra radbrytningar.

```
Denna länk kommer att brytas om den inte får plats.<p>
<nobr>Men den här kommer att fortsätta på samma rad.</nobr>
```

Figur 3.15 b och c
Risken med <nobr> är att en text blir så lång att den inte får plats i fönstret. <nobr> bör alltså användas med försiktighet.



Figur 3.15 b

Figur 3.15 c

Kommentarer i HTML-dokumentet

Det finns en särskild kod som låter dig skriva kommentarer i HTML-dokumentet. Så här ser den ut:

```
<!-- Detta är en kommentar -->
```

Allt det som står mellan ”<!--” och ”-->” ignoreras av webläsaren och syns alltså inte på hemsidan. Det fungerar lika bra om man tar bort bindestrecken och bara skriver ”<!” och ”>”.

Kommentarer är praktiska att använda om flera personer jobbar med samma HTML-dokument. Små förklarande texter gör det lättare för den som inte själv skapat dokumentet att sätta sig in i hur det är upplagt.

Att felsöka ett HTML-dokument

Ett mycket viktigt användningsområde för kommentar-koden är felsökning. Genom att sätta ett utropstecken efter första ”<”-tecknet på alla koder som du misstänker inte fungerar förvandlas koderna till kommentarer som webläsaren ignorerar.

Du kan till exempel tillfälligt ta bort fet stil ur en rad genom att lägga till utropstecken: Denna text är <!b>FET!<!/b>. Texten kommer nu inte längre att visas som fet. Tänk på att både och avslutningskoden måste göras till kommentarer. När du sedan vill aktivera koden igen är det bara att ta bort utropstecknet.

Om en mer komplicerad sida inte beter sig som du vill kan du förvandla en kod i taget till en kommentar och kontrollera sidan efter varje förändring. På så sätt eliminerar du steg för steg möjliga felkällor tills du hittar felet i kodningen.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Förkortningarna till höger anger med vilka webläsare koderna och attributen fungerar.
ns = Netscape Navigator, ie = Internet Explorer. Siffrorna anger version.

HEMSIDANS BESTÅNDSDELAR

HTML-kod	Webläsare
<html>…</html>	alla
Skapar hela HTML-dokumentet.	
<head>…</head>	alla
Skapar dokumentets huvud.	
<title>…</title>	alla
Innehåller hemsidans titel. Ska ligga i huvuddelen.	
<body>…</body>	alla
Skapar HTML-dokumentets kroppsdel.	

KODER SOM FORMATERAR TEXT

…	alla
Skapar fet stil.	
<blink>…</blink>	ns2-4
Gör så att texten blinkar.	
…	alla
Gör texten kursiv.	
<pre>…</pre>	alla
Texten återges med fast bredd och de radbrytningar som gjorts i HTML-dokumentet behålls.	
<tt>…</tt>	alla
Gör så att texten skrivs med ett typsnitt som har fast bredd, oftast Courier.	
<u>…</u>	alla
Skapar understruken text. Kan utseendemässigt förväxlas med klickbara länkar och bör därför användas sparsamt.	

KODER SOM DELAR UPP TEXTEN VERTIKALT

<hr />	alla
Skapar en horisontell linje.	
Attribut till <hr />	
align=*”left, right”* eller *“center”*	alla
Gör linjen vänsterställd, högerställd eller centrerad.	
noshade	alla
Tar bort linjens skugga.	
size=”*värde”*	alla
Ändrar linjens tjocklek. Värdet uttrycks i pixlar.	

HTML-kod	Webläsare
width=*"värde"* eller *"värde%"*	alla
Ändrar linjens bredd, antingen mätt i pixlar eller som procent av webläsarens fönster.	
<h*värde*>…</h*värde*>	alla
Skapar en rubrik. Värdet kan vara mellan 1 (störst) och 6 (minst).	
 	alla
Skapar en radbrytning.	
…	alla
Skapar nytt stycke.	
**Attribut till **	
align=*"left"*, *"right"* eller *"center"*	alla
Vänsterjusterar, högerjusterar eller centrerar efterföljande text.	
KODER SOM ÄNDRAR STORLEK PÅ TEXTEN	
…	alla
Attribut till <font>	
size=*"värde"* eller *"+/–värde"*	alla
Sätter en ny storlek på efterföljande text. Värdet kan vara mellan 1 (minst) och 7 (störst). +/- värde ändrar storlek i angivet antal steg i förhållande till <basefont>-värdet.	
face="*typsnitt1*, *typsnitt2* (osv)"	alla
<basefont />	alla
Attribut till <basefont>	
size=*"värde"*	alla
Ändrar storlek på standardtext. Kan vara 1 (minst) till 7 (störst).	
KODER SOM JUSTERAR TEXT OCH BILD	
<blockquote>…</blockquote>	alla
Skapar ett indrag av både höger och vänster marginal. Bra för att skilja citat från övrig text.	
…	alla
Centrerar efterföljande text och bilder.	
<div align=*"left"*, *"right"* eller *"center"*>…	ie3-4·ns2-4
Gör efterföljande text vänsterställd, högerställd eller centrerad.	
Övriga koder	
<nobr>…</nobr>	alla
Hindrar webläsaren att göra radbrytningar.	
<!-- *kommentartext* -->	alla
Kommentar. Allt mellan "<!--" och "-->" ignoreras av webläsaren. Inga "<" eller ">" får finnas inuti kommentaren. Bindestrecken kan utelämnas.	

4 ATT KOPPLA IHOP SIDORNA: Hyperlänkar

Möjligheten att skapa länkar från en websida till en annan är en av huvudpoängerna med World Wide Web. I det här kapitlet lär du dig hur det går till att skapa sådana länkar, och hur du låter dina besökare skicka e-post till dig med ett enkelt musklick. Dessutom ska vi titta på hur filsystemet på en dator fungerar och dra en och annan slutsats av det.

När du klickar på en länk i ett HTML-dokument utnyttjar du en av de funktioner som var själva orsaken till att World Wide Web skapades: hyperlänken. Kanske har du tänkt på att det finns flera olika typer av länkar. Dels den vanligaste sorten som leder till ett nytt HTML-dokument. Dels sådana som leder till en fil som måste laddas ned till din hårddisk för att kunna användas, eller som kanske måste läsas av ett insticksprogram till din webläsare.

Det finns också den typ av länkar som leder till ett så kallat ankare i ett och samma dokument, något som kan vara mycket snabbare och effektivare än att hoppa mellan flera olika HTML-dokument. Slutligen kan en länk tillåta användaren att skicka e-post direkt från en hemsida. I det här kapitlet kommer vi att gå igenom hur de olika länkvarianterna fungerar och kodas.

Länkens uppbyggnad

Alla länkar inleds med koden <a> och avslutas med . Den text eller den bild som placeras mellan <a> och blir den klickbara länk som syns på hemsidan. För att markera att det rör sig om en länk brukar texten bli blåfärgad och understruken, medan bilder ofta får en blå ram.

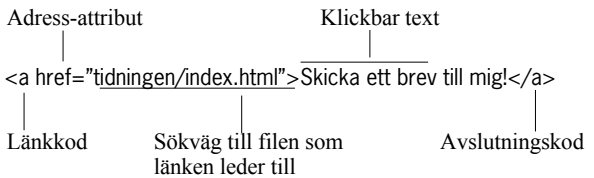
Namnet ”a” kommer från engelskans *anchor*, ankare. <a> kan nämligen också användas för att skapa ankare i ett HTML-dokument. Vi återkommer till skillnaden mellan ankare och länkar längre fram i kapitlet.

Adressen

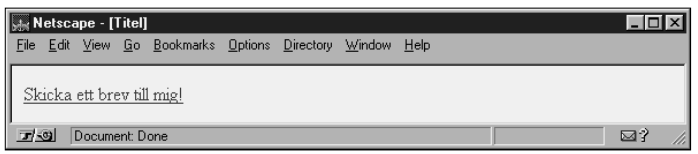
En länk fungerar inte utan en länkadress. Den specificerar du med hjälp av attributet href=”*adress”* (från engelskans *Hypertext REFerence*, hypertextreferens).

Exempel 4.1 a visar en typisk länk med alla dess olika delar. Hur länken ser ut i webläsaren ser du i figur 4.1 b.

Exempel 4.1 a
En hyperlänk med alla sina komponenter.



Figur 4.1 b
Och så här ser den klickbara länken ut i webläsarens fönster.

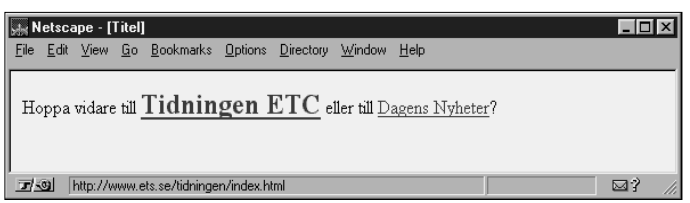


Texten mellan <a> och kan behandlas som vilken annan text som helst. Det går till exempel bra att ändra storlek och stil. I exempel 4.2 a har en av länktexterna gjorts fet och dessutom större än den övriga texten. Figur 4.2 b visar hur resultatet ser ut i en webläsare.

Exempel 4.2 a
En klickbar länk kan ändra utseende…

```
Hoppa vidare till
<a href="http://www.etc.se/tidningen/index.html"><font
size="5"><b>Tidningen ETC</b></font></a> eller till <a
href="http://www.dn.se">Dagens Nyheter</a>?
```

Figur 4.2 b
…som vilken text som helst.



Att länka två dokument till varandra

För att demonstrera hur två olika HTML-dokument länkas till varandra ska vi titta på ett exempel. Öppna ett malldokument och knappa in koden i exempel 4.3 i dokumentets <body>-del. Spara sedan sidan under namnet ”spelmeny.html”.

Exempel 4.3
Knappa in koden och spara dokumentet som ”spelmeny.html”.

```
<h1>Bra datorspel</h1>
Spelföretaget Maxis har gjort många bra simulatorspel. Välj det
spel du vill veta mer om:

<p><a href="tower.html">Sim Tower</a></p>
<p>Sim Ant</p>
<p>Sim City</p>
```

Viktigt.
De två dokumenten ”spelmeny.html” och ”tower.html” måste ligga i samma katalog. Annars kan inte webläsaren hitta fram till rätt dokument när du klickar på länkarna. Nästa avsnitt handlar just om hur man anger en sökväg i sina länkar.

Dokumentet skulle egentligen ha innehållit tre länkar, men eftersom tanken bara är att demonstrera principen nöjer vi oss med att länken till Sim Tower fungerar.

Nu måste vi skapa dokumentet som länken leder till: ”tower.html”. Öppna därför ett nytt malldokument och skriv in koden i exempel 4.4 (du kan ju byta ut texten mot något roligare om du vill). Spara dokumentet som ”tower.html” i samma katalog (viktigt) som ”spelmeny.html”. Öppna slutligen ”spelmeny.html” med din webläsare. Resultatet bör se ut ungefär som figur 4.5. Provklicka dig fram och tillbaka mellan sidorna. Om allt gått som det ska har du nu skapat dina första två länkar.

Exempel 4.4
Knappa in koden och spara dokumentet som ”tower.html”.

```
<h1>Sim Tower</h1>
I Sim Tower får du som spelare uppleva hur det är att administrera
en skyskrapa.
<p><a href="spelmeny.html">Tillbaka till spelmenyn</a></p>
```

Figur 4.5
”Spelmeny.html”. När du klickar på orden ”Sim Tower” bör du komma till dokumentet ”tower.html”.

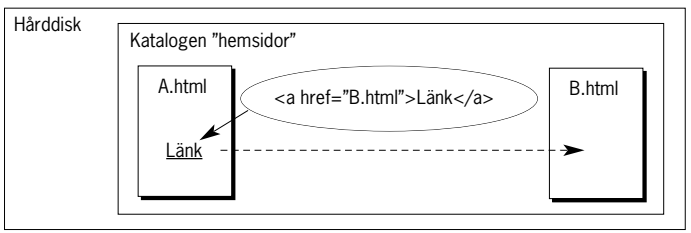
Sökvägar till dokument

Den rad som kommer efter ”href=” i en hyperlänk brukar kallas *sökväg*. Sökvägen talar om för datorn vad dokumentet som länken leder till heter, och var det finns. Men om du tittar noga på länkarna i exempel 4.3 och 4.4 märker du att sökvägarna är väldigt korta; de består faktiskt bara av namnet på filerna: ”tower.html” respektive ”spelmeny.html”.

Anledningen till att sökvägarna inte behöver vara längre är dels att vi valt att använda så kallade *relativa sökvägar*, dels att båda dokumenten ligger i samma katalog (eller ”mapp” som samma sak heter på en Macintosh). Vi hade också kunnat använda så kallade *absoluta sökvägar*.

Relativa sökvägar

Vi återkommer till dessa längre fram i kapitlet.
När man använder en relativ sökväg i en hyperlänk utgår man alltid från den katalog som innehåller dokumentet med länken, och talar sedan om för webläsaren hur den därifrån ska hitta fram till dokument som länken leder till.



Figur 4.6
När två dokument ligger i samma katalog behöver man inte tala om mer än dokumentnamnet för att länkar mellan dem ska fungera.

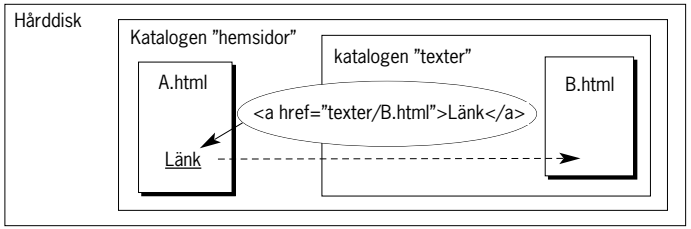
Titta på figur 4.6. De stora rutorna representerar kataloger på en hårddisk. Dokumenten A.html och B.html ligger båda i en katalog som heter ”hemsidor”. A.html innehåller en länk till B.html. Denna länk behöver inte innehålla något annat än namnet på länkdokumentet, eftersom båda dokumenten ligger i samma katalog. Den korrekta länken blir därför:

```
<a href=”B.html”>Länk</a>
```

I nästa exempel (figur 4.7) ligger dokument A.html kvar i katalogen ”hemsidor”, medan B.html flyttats till underkatalogen ”texter”. Man brukar säga att B.html ligger *en nivå längre ned* i katalogstrukturen än A.html. För att länken till B.html ska fungera måste vi nu tala om för webläsaren var B.html finns. Detta gör vi genom att skriva namnet på underkatalogen följt av ett snedstreck framför namnet på HTML-dokumentet. Den korrekta hyperlänken till B.html ser alltså ut så här:

```
<a href=”texter/B.html”>Länk</a>
```

Figur 4.7
Den här länken leder till underkatalogen ”texter”. Sökvägen måste då innehålla både underkatalogens och HTML-dokumentets namn.



Katalogstruktur.
Det sätt på vilket en dator organiserar filer på sin hårddisk kallas för en filstruktur. Filstrukturen ser ungefär likadan ut oavsett vilken dator du använder: Enskilda filer organiseras i kataloger, eller ”mappar” som de kallas på en Macintosh, och varje katalog kan i sin tur innehålla underkataloger.

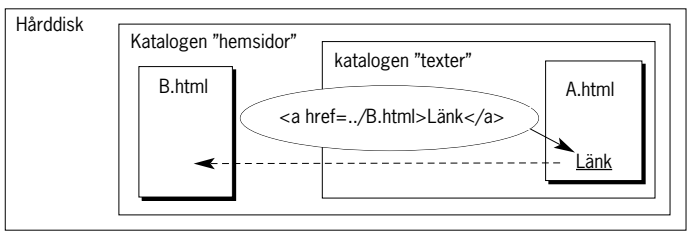
Om ”B.html” legat ytterligare en eller flera nivåer under ”A.html” hade vi bara lagt till namnet på underkatalogerna i tur och ordning. Då hade länken kanske sett ut så här:

```
<a href=”texter/recept/soppor/B.html”>Länk</a>
```

Nu vänder vi på alltihop. I figur 4.8 har A.html och B.html bytt plats med varandra. För att länken till ”B.html” nu ska fungera måste vi alltså få webläsaren att röra sig ett steg *uppåt* i katalogstrukturen, från underkatalogen ”texter” till katalogen ”hemsidor”. Detta åstadkommer vi genom att inleda sökvägen med två punkter följt av ett snedstreck (../). Den korrekta sökvägen till ”B.html” blir alltså:

```
<a href=”../B.html”>Länk</a>
```

Figur 4.8
Den här länken leder ett steg uppåt i katalogstrukturen. Detta åstadkoms med två punkter följt av ett snedstreck (../).



Webläsaren hoppar en nivå upp för varje omgång punkter och snedstreck i sökvägen. Om ”B.html” legat ytterligare en eller flera nivåer över ”A.html” hade vi alltså bara lagt till fler punkter och snedstreck. Länken skulle då kunna se ut så här:

```
<a href=”../../../B.html”>Länk</a>
```

Hur du slipper sökvägarna

Om du tycker att det är krångligt med sökvägar finns det ett bra sätt att slippa bry sig om dem: lägg alla dina bilder och HTML-dokument i en enda katalog. Fördelen är att dina länkar då aldrig behöver innehålla något annat än namnet på dokumentet som länkarna leder till (figur 4.6). På så sätt minimerar du dessutom risken att råka skriva fel sökväg, med icke

fungerande länkar som resultat. Detta brukar även den mest rutinerade HTML-snickare annars råka ut för, åtminstone någon gång.

Nackdelen med att förvara allt material i en katalog är naturligtvis att den blir väldigt stor och rörig om du bygger upp ett stort system med många enskilda HTML-dokument och bilder.

Om du trots allt vill dela upp ditt material i flera kataloger är relativa sökvägar nästan alltid det enklaste och bästa sättet att tala om var länkfilerna finns. Tyvärr finns det ett par tillfällen då de inte fungerar:

- I en så kallad *serverbaserad bildkarta.* Det är en sorts klickbar bild som du kommer att få lära dig använda i kapitel 7.
- När en länk leder till en annan dator på Internet.

Nästa avsnitt handlar om den sistnämnda sortens länkar.

Sammanfattning av relativa sökvägar

Nedan följer en sammanfattning av hur du dirigerar webläsaren till rätt dokument med hjälp av relativa sökvägar.

href=”dokument.html”
”dokument.html” ligger i samma katalog som dokumentet med länken.

href=”texter/dokument.html”
”dokument.html” ligger i underkatalogen ”texter”, som i sin tur ligger i samma katalog som dokumentet med länken.

href=”texter/lyrik/dokument.html”
”dokument.html” ligger i katalogen ”lyrik”, som är en underkatalog till ”texter”, som i sin tur ligger i samma katalog som dokumentet med länken.

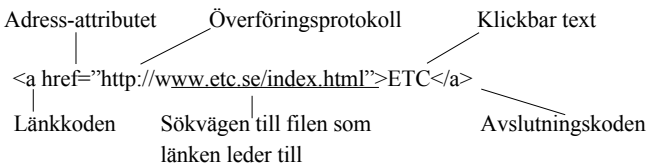
href=”../dokument.html”
”dokument.html” ligger i en katalog som befinner sig en nivå högre upp än katalogen som innehåller dokumentet med länken.

href=”../../dokument.html”
”dokument.html” ligger i en katalog som befinner sig två nivåer högre upp än katalogen som innehåller dokumentet med länken.

href=”../lyrik/dokument.html”
”dokument.html” ligger i katalogen ”lyrik”, som i sin tur befinner sig en nivå högre upp än katalogen som innehåller dokumentet med länken.

Absoluta sökvägar

Om en länk ska leda till en annan dator på Internet måste man använda en så kallad *absolut sökväg*. En sådan innehåller inte bara namnet på filen som länken leder till, utan också adressen till den dator i vilken filen lagrats. Figur 4.9 visar en typisk länk med absolut sökväg.



Figur 4.9
En länk till Internet. Den absoluta sökvägen skiljer sig från den relativa genom att ett överföringsprotokoll måste anges, och att adressen till datorn som filen ligger på också måste finnas med.

En länk med absolut sökväg skiljer sig som synes inte så mycket från en som använder relativ sökväg. Största nyheten är bokstäverna ”http://”. Detta betyder *HyperText Transfer Protocol* och är helt enkelt en uppsättning regler för hur hemsidor ska transporteras mellan Internets alla datorer. Sådana regler brukar kallas *protokoll.* Ett annat vanligt protokoll är FTP (*File Transfer Protocol),* med vilket man kan hämta filer i särskilda filarkiv. På World Wide Web är det dock nästan alltid protokollet HTTP som gäller.

Nästa nyhet är adressen till datorn: www.etc.se. Alla datorer på Internet har en unik adress, som kan uttryckas dels som siffror, dels som bokstäver. Siffermotsvarigheten till www.etc.se är 195.7.66.223, och länken fungerar därför lika bra om vi skriver:

ETC.

Efter adressen till datorn följer namnet på dokumentet som länken leder till, ”index.html”. Om detta dokument legat i en särskild katalog hade namnet på denna också funnits med. Länken skulle då kunna se ut så här:

ETCs arkiv

Prova att skapa ett HTML-dokument som innehåller länkarna ovan. Tänk på att du måste vara uppkopplad mot din Internetleverantör för att kunna prova länkarna. De leder ju utanför din egen dator.

Länkar inom ett dokument

Hittills har vi låtit alla länkar gå från ett HTML-dokument till ett annat. Men en länk kan också leda från en plats till en annan i ett och samma dokument. På så sätt kan man på ett överskådligt sätt få in mycket information på en hemsida. Om informationen istället sprids på flera olika hemsidor tar den dessutom totalt sett längre tid att ladda ned, eftersom en webläsare tar en stund på sig att lokalisera, beställa och ladda ned varje enskild sida. Om all information samlas på en enda hemsida behöver nedladdningsprocessen bara klaras av en gång.

Länkar inom en hemsida består av två delar: själva länken och ett så kallat *ankare* dit länken leder. Båda skapas med hjälp av länkkoden <a>, men använder sig av olika attribut.

Länkdelen

Så här ser länkdelen ut:

```
<a href="#ankarnamn">Klickbar text</a>
```

Den enda nyheten jämfört med de länkar vi tidigare gått igenom är att hela sökvägen bara består av ett enda ord: ankarnamnet (vi återkommer strax till vad detta är). För att webläsaren ska förstå att länken inte leder till ett vanligt HTML-dokument ska ankarnamnet dessutom föregås av ett brädgårdstecken (#).

Ankardelen

Ankardelen ska placeras på den punkt i HTML-dokumentet dit du vill att webläsaren ska hoppa när någon klickar på länken. Så här är ankaret konstruerat:

```
<a name="ankarnamn">Lämplig text eller bild</a>
```

Den stora skillnaden är som synes att attributet href nu bytts ut mot ett helt nytt attribut: name="*ankarnamn*".

Namnet kan i princip se ut hur du vill, men undvik svenska tecken och andra specialtecken. Namnet måste det sättas mellan citationstecken: name="*ankarets namn*".

För att visa hur det hela fungerar i praktiken ska vi göra ett experiment. Skapa ett nytt HTML-dokument och knappa in koden i exempel 4.10 a (byt gärna ut texten mot något roligare om du vill). Spara dokumentet under namnet "spel.html". Namnet är viktigt, för vi återkommer till dokumentet senare.

Exempel 4.10 a
"spel.html" innehåller fyra interna länkar. Tre går till spelrubrikerna, medan den fjärde går tillbaka till den översta rubriken "Bra datorspel".

```
<a name="Spelmeny"><h1>Bra datorspel</h1></a>
Spelföretaget Maxis har gjort många bra simulatorspel. Välj det
spel du vill veta mer om:<p>
<a href="#Spel 1">Sim Tower</a><br />
<a href="#Spel 2">Sim Ant</a><br />
<a href="#Spel 3">Sim City</a>
<hr />
<a name="Spel 1"><h1>Sim Tower</h1></a>
I Sim Tower får du som spelare uppleva hur det är att administrera
en skyskrapa. Till en början är huset bara några våningar högt,
men om du gör ett bra jobb slutar du på 100 våningar och tusen-
tals invånare.</p>
<a href="#Spelmeny">Tillbaka till spelmenyn</a><p>
<a name="Spel 2"><h1>Sim Ant</h1></a>
I Sim Ant får spelaren styra en elektronisk myrstack.
<a href="#Spelmeny">Tillbaka till spelmenyn</a></p><p>
<a name="Spel 3"><h1>Sim City</h1></a>
Klassikern i Simserien. Spelaren är borgmästare för en stad som
utvecklas från håla till världsmetropol.
<a href="#Spelmeny">Tillbaka till spelmenyn</a></p>
```

Svenska tecken. Om de svenska tecknen (å, ä och ö) ser konstiga ut på din hemsida kan det bero på att du använder Macintosh. Bläddra i så fall fram till kapitel 18, där lösningen på problemet finns.

Figur 4.10 b och c visar hur kodningen i exempel 4.10 a ser ut i webläsarens fönster. När du klickar på länkarna kommer webläsaren att hoppa ned så att de ord du utsett till ankare hamnar överst i fönstret. Vid varje avdelning finns också en länk som leder till det allra översta ankaret, "Spelmeny". Det är bra att på detta sätt låta användaren enkelt klicka sig tillbaka till sidans början, särskilt om sidan är mycket lång. Det går ju betydligt fortare att förflytta sig via länkarna än med rullisterna.

Om du gör fönstret så högt att hela hemsidan får plats på en gång händer ingenting då du klickar på en intern länk.

Figur 4.10 b och c
När du klickar på interna länkar hoppar webläsaren ned till respektive ankare (lilla bilden). För att användaren snabbt ska kunna ta sig tillbaka till början finns det vid varje ankare en länk som leder till toppen av sidan.

Ankare i andra dokument

Det finns ett sätt att låta en länk leda till ett ankare som finns i ett annat HTML-dokument.

Om vi till exempel vill skapa en länk som går till dokumentet ”spel.html” (exempel 4.10 a) och dessutom vill att rubriken ”Sim Ant” ska hamna överst i fönstret då hemsidan visas, är det bara att lägga till ett brädgårdstecken och det rätta ankarnamnet direkt efter dokumentnamnet i sökvägen. Så här ska det se ut:

```
<a href="spel.html#Spel 2">Spel</a>
```

När användaren klickar på länken kommer han eller hon inte bara att få upp ”spel.html” i fönstret. Webläsaren kommer dessutom att leta upp ankaret ”Spel 2” (som finns vid rubriken ”Sim Ant”) och placera denna rubrik överst.

Om inte länken fungerar

Om dina länkar inte fungerar är sannolikheten stor att du glömt eller felplacerat brädgårdstecknet. Det måste finnas med i alla länkar som leder *till* ett ankare. Det ska däremot inte finnas med när du ger ankaret dess namn med <a name=”*ankarnamn”*>. Det är också väldigt lätt att glömma citationstecken (”) i länknamn innehållande mellanslag, och då fungerar inte länken. Kontrollera alltså noga att alla namn som ska ha citationstecken verkligen har ett i början och ett i slutet.

Kontrollera också att du använder exakt samma namn i länken som i ankaret. Om ankaret börjar med stor bokstav måste även namnet i länken göra det. ”Ankare nummer ETT” är alltså inte detsamma som ”ankare nummer ett”. Detta gäller för övrigt också namn på bilder och alla andra filer. För att minimera risken att skriva fel kan det vara idé att hålla sig till små bokstäver i alla ankarnamn och filnamn. Om du planerar att göra stora dokument med många länkar kan ett konsekvent användande av små tecken spara mycket felsökningstid.

Du bör dessutom alltid noga kontrollera att sökvägarna verkligen är riktigt inskrivna, särskilt om du nyligen döpt om ett eller flera dokument. Då måste ju sökvägen i alla länkar som leder till dokumentet också ändras. Om du ska sköta riktigt stora system av hemsidor kan det vara idé att skaffa ett program som hjälper dig att hålla koll på alla länkarna och ändrar automatiskt samtliga sökvägar när du döper om eller flyttar ett dokument.

Viktigt.
Vissa versioner av Netscape Navigator och Internet Explorer saknar inbyggd posthantering. I dessa läsare fungerar inte koden till höger. Skriv därför alltid ut din e-postadress så att besökarna kan skicka post på vanligt sätt.

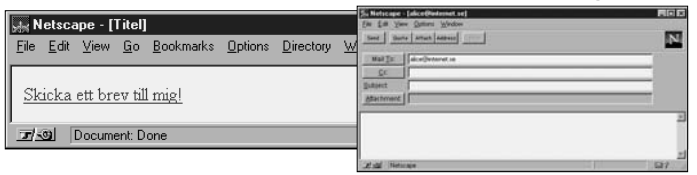
Att skicka brev från hemsidan

Om dina besökare med ett enkelt musklick kan skicka brev till dig direkt från hemsidan kommer du att få mer post än om de manuellt måste knappa in din e-postadress. Tricket åstadkoms med hjälp av en vanlig länk och attributet MAILTO. Så här ser koden ut:

```
<a href="mailto:epost@adress">Klickbar text</a>
```

Om personen som brevet ska skickas till har e-postadressen alice@internet.se ska länken se ut så här:

```
<a href="mailto:alice@internet.se">Skicka ett brev till mig!</a>
```



Figur 4.11
När besökaren klickar på länken...

Figur 4.12 (höger)
...dyker ett brev med redan ifylld adress upp. Det är bara att fylla i texten och skicka iväg det.

Figur 4.11 visar hur länken ser ut i webläsarens fönster. När användaren klickar på texten skapas automatiskt ett nytt brev där adressen redan är ifylld (figur 4.12). Sedan är det bara att skriva brevet och skicka iväg det. Prova att skapa en MAILTO-länk med din egen e-postadress.

Precis som med vanliga länkar går det bra att byta ut texten mellan och mot en bild. Den blir då klickbar och markeras med blå ram.

Länkar i nya fönster

Ibland kan det vara en fördel om användaren hindras att surfa ifrån din hemsida så fort han eller hon klickar på en länk. En metod är att låta webläsaren öppna ett helt nytt fönster åt länken. Detta görs med hjälp av target=”*fönsternamn”*, som är ett attribut till <a>. Följande kod skapar ett nytt fönster med namnet ”etc” och i det visas dokumentet som länken leder till:

```
<a href="http://www.etc.se/tidningen/index.html"
target="etc">ETC</a>
```

Om du inte är intresserad av att skicka mer än en länk till ett fönster behöver du inte namnge det. Target=”_blank” (namnet ska börja med en understrykning) åstadkommer ett nytt, namnlöst fönster.

Target är en viktig kod när man jobbar med rutor (*frames*). Det lär du dig mer om i kapitel 10.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Samtliga nedanstående koder fungerar med Netscape Navigator 2.0-4.0 (eller senare) och Internet Explorer 2.0-4.0 (eller senare).

KODER SOM SKAPAR LÄNKAR

HTML-kod

<a>...

Används för att skapa en länk eller ett ankare.

Attribut till <a>

href=*"sökväg#ankarnamn"*

Skapar en länk till ett annat dokument och/eller ett ankare.

mailto:*namn@adress*

Skapar en länk till ett föradresserat brev.

target=*"fönsternamn"*

Öppnar länken i ett nytt fönster.

name=*"ankarnamn"*

Skapar ett ankarnamn som länkar kan leda till.

id=*"id-namn"*

Skapar ett id.

5 ORGANISERA DIN HEMSIDA:

Listor i HTML

Om du ska redovisa mycket information på din hemsida är det ofta praktiskt att använda olika typer av listor. Listfunktionen är mycket välutvecklad i HTML, och i det här kapitlet går vi igenom alla de listalternativ du har att välja mellan.

Det finns många olika sätt att redovisa text i punktform, definiera ord och numrera textavsnitt med hjälp av HTML. Denna utmärkta kapacitet för organisation och hierarkisk presentation av text är mycket typisk för HTML, och kanske också för den forskarvärld som språket kommer från. Det märks att HTML skapades av folk som inget annat ville än att publicera forskningsrapporter på sina hemsidor. Det fanns många metoder för att presentera statistik på en hemsida långt innan det gick att till exempel ändra tjockleken på en bildram.

Lyckligtvis finns det många tillfällen då även vi som inte planerar att doktorera har användning för listor. Även kortare texter kan bli mycket lättare att läsa om en del av materialet redovisas i punktform. Dessutom skapar listorna luft på hemsidan, som annars lätt kan kännas ihopklämd när den fylls med text.

Det finns tre huvudtyper av listor att välja mellan:

- Onumrerade listor som redovisar innehållet i punktform
- Numrerade listor
- Definitionslistor

Av de olika listsorterna är nog definitionslistan den minst använda. Den används huvudsakligen för att definiera ord, men i slutet av kapitlet ska vi titta på hur en definitionslista faktiskt också kan användas för att skapa en rejäl vänstermarginal på din hemsida.

Onumrerade listor

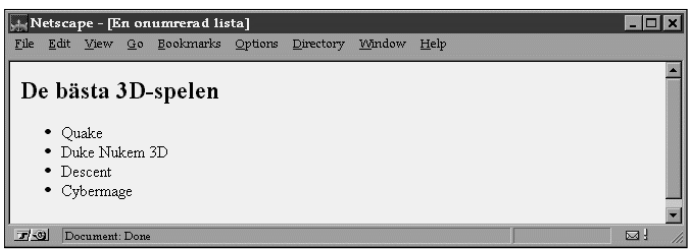
Den onumrerade listan visas i punktform. Koden som inleder ser ut så här: <ul> (efter engelskans *unordered list*). Därefter måste varje punkt i listan föregås av koden <li> (*list item*). Varje gång webläsaren stöter på ett nytt <li> tolkas detta som att en ny punkt i listan ska inledas. Hela listan avslutas med slutkoden . Exempel 5.1 a och figur 5.1 b visar en enkel onumrerad lista.

I kodningen är vissa rader indragna med hjälp av TAB-tangenten. Detta påverkar inte alls kodens funktion men gör den lättare att läsa, särskilt när listorna blir mer komplicerade. Prova gärna att själv knappa in en lista, och experimentera med att öka antalet rader.

Exempel 5.1 a
En onumrerad lista.

```
<h2>De bästa 3D-spelen</h2>

<ul>
  <li>Quake</li>
  <li>Duke Nukem 3D</li>
  <li>Descent</li>
  <li>Cybermage</li>
</ul>
```

Figur 5.1 b
Resultatet av kodningen i exempel 5.1 a.



Flera listor i varandra

Viktigt.
Koden blir lätt rörig när flera listor ingår i varandra. Var noga med att alla <ul> verkligen avslutas med . Annars kan listan haverera.

Det går bra att låta en eller flera mindre listor ingå i en större lista. Det åstadkommer du genom att ersätta en <li> med en ny <ul>:

```
<ul> <! Här startar huvudlistan>
  <li>3D-spel</li>
        <ul><li>Quake</li><li>Doom</li></ul>
  <li>Descent</li>
  <li>Cybermage</li>
</ul>
```


Numrerade listor

En annan listtyp är den numrerade. Koden som startar listan heter <ol> (efter engelskans *ordered list*), och precis som i den onumrerade listan inleds varje listobjekt med koden <li>. Skillnaden är att både <ol> och <li> nu har attributet TYPE=*värde*. Värdet bestämmer på vilket sätt listans punkter ska numreras. Här är alla de möjliga värdena:

Värde	**Listan numreras med...**
type=”A”	stora bokstäver
type=”a”	små bokstäver
type=”I”	stora romerska siffror
type=”i”	små romerska siffror
type=”1”	siffror

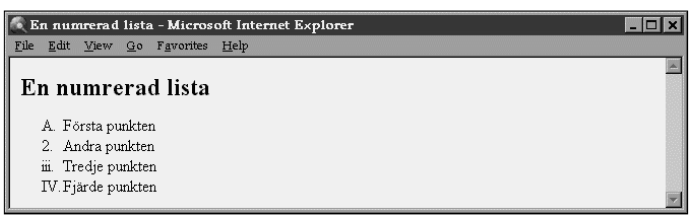
Om attributet type=”*värde*“ används tillsammans med <ol> kommer alla punkterna i listan att numreras på det angivna sättet. Om attributet i stället används tillsammans med ett enskilt <li> påverkas bara den individuella raden.

Exempel 5.2 a och figur 5.2 b visar hur <ul> används och hur den numrerade listan ser ut i webläsaren.

Exempel 5.2 a
En numrerad lista där alla de fyra punkterna numreras på olika sätt. Om exempelvis attributet type=”1” istället placerats tillsammans med <ol> hade hela listan numrerats med siffror.

```
<h2>En numrerad lista</h2>
<ol>
  <li type=”A”>
  Första punkten</li>
  <li type=”1”>
  Andra punkten</li>
  <li type=”i”>
  Tredje punkten</li>
  <li type=”I”>
  Fjärde punkten</li>
</ol>
```

Figur 5.2 b
Resultatet av kodningen i exempel 5.2 a.



Om du vill att din numrerade lista ska starta på en särskild siffra går det bra. Lägg bara till attributet start=”*värde*“ till -koden. <ol start=”5”> skapar alltså en numrerad lista där första raden har nummer fem. start-värdet ska alltid anges med vanliga siffror, och ändras sedan automatiskt till ”E”, ”e”, ”V”, eller ”v” beroende på vilken listtyp du väljer. Det finns också ett särskilt attribut för att hoppa framåt eller bakåt mitt inne i listan. Skriv då value=”*värde*“ tillsammans med <li> på den rad där hoppet ska ske. Värdet är i det här fallet den siffra du vill att raden ska ha. Den fortsatta uppräkningen sker sedan från detta nummer.

Definitionslistan

Listtypen som återstår kallas för definitionslista och fungerar annorlunda än de andra två varianterna. En definitionslista börjar visserligen som vanligt med koden <dl> (efter engelskans *definition list*) och avslutas med </dl>. Men där emellan finns två olika koder: Orden som ska definieras föregås av <dt> (efter engelskans *definition list title*) och varje definition med koden <dd> (efter *definition list data*). Om det låter krångligt kanske exemplet nedan får det hela att klarna.

Exempel 5.3 a
Definitionslistan. Orden som ska definieras markeras med <dt> och definitionerna med <dd>.

```
<h2>Bra 3D-spel</h2>

<dl>
  <dt>Quake</dt>
  <dd>Den utmärkta uppföljaren till Doom.<br />
  Spelet släpptes som shareware i maj 1996.</dd>
  <dt>Descent</dt>
  <dd>3D-spel i rymdmiljö.</dd>
  <dt>Cybermage</dt>
  <dd>Doom-variant som även innehåller inslag av rollspel.</dd>
</dl>
```

Figur 5.3 b
Resultatet. Lägg märke till att varje definition som markerats med <dd> i kodningen ovan får ett rejält indrag.

Fixa en marginal med definitionslistan

Det finns ytterligare ett sätt att utnyttja koden för definitionslista: till att ge hemsidan en vänstermarginal. Om du tittar i vilken bok eller tidning som helst märker du snart att texten aldrig börjar ända ute vid sidans vänstra kant. Där ska alltid finnas minst någon centimeter tomt utrymme för att ögat lätt ska kunna hitta rätt rad under läsning.

När HTML skapades tog man ingen hänsyn till detta, varför texten på din hemsida alltid kommer att hamna längst ute till vänster om du inte gör något åt det (se figur 5.4 a).

Ett enkelt sätt att skapa en vänstermarginal åt hela dokumentet på en gång är att inleda med att starta en definitionslista, hoppa över definitionsordet och låta hela hemsidans innehåll bli en enda definition (se figur 5.4 b). Resultatet blir en infällning som gör din sida både snyggare och mer lättläst (se figur 5.4 c).

Du behöver inte ens avsluta definitionslistan när HTML-dokumentet är slut. Det fungerar ändå.

Figur 5.4 a
I normala fall hamnar texten på en hemsida ända ute vid vänsterkanten. Raderna blir då svåra att följa för ögat.

Figur 5.4 b
Om texten föregås av koderna <dl><dd> uppfattas hela hemsidan som en enda definition och blir därför indragen.

Figur 5.4 c
Resultatet blir en marginal som gör texten lättare att läsa.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Samtliga nedanstående koder fungerar med Netscape Navigator 2.0-4.0 (eller senare) och Internet Explorer 2.0-4.0 (eller senare).

KODER SOM SKAPAR LISTOR

HTML-kod

…
Skapar en onumrerad lista. Varje rad i listan markeras med <li>....

…
Skapar en numrerad lista.

**Attribut till **

type=”A”, “a”, “I”, “i” eller “1”
Bestämmer om listan ska numreras med stora bokstäver (A), små bokstäver (a), stora romerska siffror (I), små romerska siffror (i) eller vanliga siffror (1). Varje rad föregås av <li>.

start=”*värde*”
Bestämmer på vilken siffra listan ska starta, om inte 1.

...
Definierar en rad i en onumrerad (<ul>) eller numrerad (<ol>) lista.

Attribut till <li> om koden används i numrerad lista

type=”A”, “a”, “I”, “i” eller “1”
Bestämmer om denna rad ska numreras med stora bokstäver (A), små bokstäver (a), stora romerska siffror (I), små romerska siffror (i) eller vanliga siffror (1).

value=”*värde*”
Används för att göra ett hopp i listan. Raden får det värde som anges, och uppräkningen fortsätter på kommande rader.

<dl>…</dl>
Skapar en definitionslista.

Innehåll i <dl>

<dt>...</dt>
Markerar ord som ska definieras.

<dd>...</dd>
Markerar definitionen.

6 GE LIV ÅT DIN HEMSIDA:

Bilder

Innan World Wide Web startade sitt segertåg över Internet var global datorkommunikation en oestetisk företeelse. Datorer kopplades ihop via gråa terminalprogram där en blinkande markör var det roligaste som fanns att titta på. Men WWW förändrade Internets utseende, och idag är de flesta överens om att det var just bilderna som gjorde att World Wide Web kunde slå igenom så fort. I det här kapitlet får du veta allt om hur man fyller en hemsida med bilder.

I kapitel 1 tittade vi på hur det går till när en amerikansk hemsida transporteras över Internet till din dator. Nu är det dags för en snabbrepetition: Först anropar webläsaren den dator där hemsidan finns. Denna sänder över ett HTML-dokument innehållande tre saker: texten, information om var på Internet bilderna som ska visas finns och information om på vilket sätt texten och bilden ska presenteras. Webläsaren börjar med att avsätta utrymme för bilderna, placerar därefter ut texten på skärmen och skickar slutligen efter bilderna, som anländer separat till din dator.

Ett HTML-dokument innehåller alltså bara bildernas adresser, aldrig bilderna i sig. Av detta följer att bilderna på en hemsida inte alls behöver finnas lagrade på samma dator som HTML-dokumentet. En och samma sida kan tvärtom innehålla bilder från flera olika datorer på Internet.

Sökvägen till en bild

I kapitel 4 såg vi hur man anger sökvägen till en länk. Att ange sökvägen till en bild fungerar precis likadant. Om bilden ligger på samma dator som HTML-dokumentet räcker det alltså med att ange en relativ sökväg som bara innehåller information om i vilken katalog bilden finns att hämta. Om

den lagras i en annan dator på Internet måste sökvägen vara absolut, och alltså även innehålla adressen till själva datorn.

HTML-koden som används för att visa bilder heter <img /> (efter engelskans *image*, bild) och har "inbyggd" avslutningskod. För att koden ska ha någon effekt måste den alltid kompletteras med attributet src (efter engelskans *source*, källa) följt av sökvägen till bilden.

En bild med absolut sökväg kan se ut så här i HTML-dokumentet:

```
<img src="http://www.etc.se/bilder/min_bild.gif" />
```

En bild med relativ sökväg kan se ut så här:

```
<img src="bilder/min_bild.gif" />
```

Mer detaljer om relativa och absoluta sökvägar hittar du i kapitlet om länkar (kapitel 4).

Olika typer av bilder

Det finns många olika sätt att lagra bilder på datorer, så kallade *bildformat.* I exemplen ovan slutar båda bildnamnen med en punkt följd av bokstäverna "gif". Denna så kallade *förlängning* av filnamnet anger vilket bildformat filen är sparad i.

Bildformat har ofta långa, krångliga namn som få kommer ihåg. Istället är det förkortningarna som används i dagligt tal: BMP, IFF, EPS, PCX, PICT, Jpeg, Gif och Png är några vanliga bildformat.

De tre sistnämnda, Jpeg (uttalas "j-pegg"), Gif (uttalas "giff") och png (uttalas "ping"), är de som används på WWW.

? Storlek på bilder.

När man talar om en bilds storlek i datorsammanhang kan man mena två saker:
* Hur stor bilden är på skärmen, vilket ofta mäts i bildskärmspunkter (pixlar).
* Hur mycket utrymme bilden tar upp då den lagras på hårddisken, vilket mäts i kilobytes, förkortat KB, eller K.

En komprimerad bild blir mindre mätt i kilobytes. De tre bildformaten som används på WWW, jpeg, png och gif, är alla komprimerade, vilket minskar deras laddningstid.

Gif-bilder

Det överlägset vanligaste bildformatet på World Wide Web är Gif-bilder, eller *Graphics Interchange Format* som bildformatet egentligen heter. Det uppfanns av det amerikanska nätverksföretaget Compuserve för tio år sedan. Gif-bildens stora begränsning är att den bara kan innehålla 256 färger. Fördelen är att Gif-bilder är *komprimerade*, vilket innebär att de lagras på ett särskilt sätt för att ta upp så lite utrymme på hårddisken som möjligt. Det innebär också att Gif-bilder går fortare att skicka över Internet än andra typer av bilder. Det finns två sorters Gif-bilder: GIF87a och GIF89a. Du har säkert sett att bilderna på många hemsidor till en början visas suddigt, för att sedan stegvis öka i skärpa. Sådana bilder har sparats i GIF89a-formatet. Tanken är att användaren snabbt ska kunna se vad bilden föreställer och bestämma om han eller hon vill vänta tills bilden är helt nedladdad, eller surfa vidare.

T Tips.

Det är mycket bättre att visa skillnaden mellan Jpeg och Gif på din datorskärm än i en tryckt bok.
På bokens hemsida hittar du därför värdefull information om hur de olika formaten fungerar, vilken typ av bilder som passar bäst för vilket format, och en kurs i konsten att göra bilderna så små och snabbladdade som möjligt. Adressen når du från http://www.etc.se/nyckeln/. Välj sedan kapitel 6 i menyn.

En annan finess med GIF89a är att bildens bakgrundsfärg kan göras genomskinlig. I slutet av kapitlet får du veta mer om vilka program som används för att spara bilder i GIF89a-format.

Png-bilder

Png-formatet (Portable Network Graphics (vissa hävdar att förkortningen står för "PNG's Not Gif")) är det format som förväntas ersätta gif-formatet. Png är ett format som liknanar gif på flera sätt men som har en mängd fördelar, dessa är några: Till skillnad från gif-formatets begränsning till 256 färger kan en png-bild innehålla tusentals färger. Png kan hantera opacitet (grader av genomskinlighet) och har en bättre komprimeringsmetod, vilket gör att bilderna blir mindre och därmed snabbare att ladda ner. Man kan dock inte göra animeringar med png-formatet.

Den främsta anledningen till utvecklingen av png-formatet är att det till skillnad från gif-fomatet är patentfritt vilket underlättar utveckling och anpassning av formatet. Png-bilder sparas med förlängningen .png.

Jpeg-bilder

Jpeg (namnet kommer från Joint Photographic Experts Group) är ett bildformat som särskilt skapats för att visa inscannade fotografier på datorskärmen. Den stora skillnaden jämfört med Gif-formatet är att Jpeg inte har några begränsningar när det gäller antalet färger. För att en bild med miljontals färger ändå ska bli så liten att den kan skickas över World Wide Web utan för långa väntetider används så kallade förstörande kompression. Det innebär att delar av informationen som ingår i bilden helt enkelt kastas bort. Tanken är att reduceringen inte ska vara synlig, men det fungerar inte alltid. För säkerhets skull kan Jpeg-bilder därför komprimeras olika mycket beroende på hur höga kraven är på bildkvalitet respektive filstorlek. En riktigt hårt komprimerad Jpeg-bild blir liten och hanterlig, men ser inte så vacker ut. Jpeg-bilder sparas med förlängningen "jpeg" eller "jpg" i filnamnet.

Övningsbild och bildkatalog

För att kunna prova koderna som vi går igenom i det här kapitlet behöver du en bild att öva med. Enklast är att gå till bokens hemsida (www.etc.se/nyckeln/) och hämta bilderna som används i exemplen. Om du har tillgång till ett grafikprogram kan du naturligtvis också göra en egen övningsbild. Se då bara till att du kan spara bilden i antingen Jpeg- png- eller Gif-format.

En tredje variant är att leta upp en lämplig övningsbild någonstans på

World Wide Web. Så här sparar du bilden till din hårddisk:

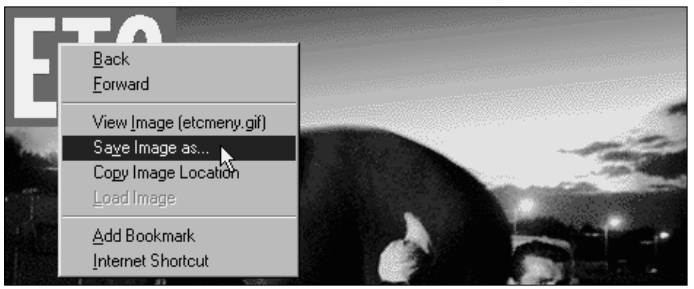
Windows:

Placera muspekaren över den bild du vill ha och håll ner höger musknapp. När den mindre menyn kommer fram klickar du på alternativet ”Save Image as...” (Netscape) eller ”Save Picture as...” (Internet Explorer) och väljer sedan en plats för bilden på din hårddisk (figur 6.1 på nästa sida).

Macintosh:

Placera pekaren över en lämplig bild och håll ner musknappen (klicka inte). I den mindre menyn som efter en kort stund dyker upp väljer du sedan ”Save this Image as...” (Netscape) eller ”Download Image to Disk” (Internet Explorer).

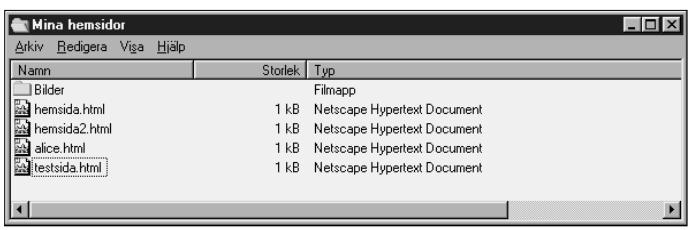
Figur 6.1
Du kan alltid spara bilder som du hittar på World Wide Web. Tänk bara på att upphovsrätten gäller även material på Internet, och att du alltså inte kan använda bilderna på din egen hemsida hur som helst.



Tänk på att lagen om upphovsrätt gäller på Internet också. Det är alltså inte fritt fram att plocka bilder från andras hemsidor och sedan publicera dem på din egen. Om du vill använda en särskild bild är det därför alltid bäst att först skicka e-post till upphovsmannen och fråga.

Passa nu på att skapa en särskild katalog för bilderna som ska användas på dina hemsidor. I exemplen utgår vi ifrån att alla bilder ligger i katalogen ”bilder” som i sin tur ligger i samma katalog som alla HTML-dokument (figur 6.2).

Figur 6.2
Det blir enkelt att hålla reda på sökvägen till dina bilder om alla ligger i samma underkatalog.

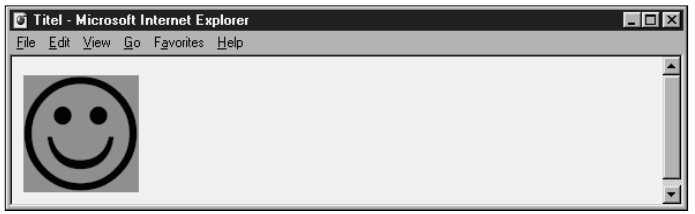


Dags att visa första bilden. Öppna ett nytt malldokument och skriv in koden i exempel 6.3 a i dokumentets <body>-del. Bilden i exemplet heter ”mellbild.gif” och ligger i katalogen ”bilder”. Spara dokumentet och öppna det i din webläsare. Resultatet bör bli ungefär som i figur 6.3 b.

Exempel 6.3 a
Så här lite kod behövs för att placera en bild på hemsidan…

```
<img src="bilder/mellbild.gif" />
```

Figur 6.3 b
…och så här ser det ut när bilden visas.



Flera upplagor av samma bild

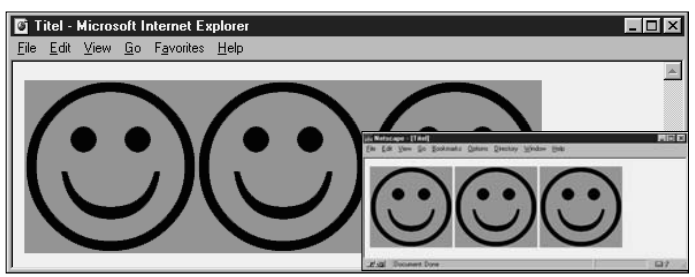
När en bild väl laddats ned kan den visas hur många gånger som helst utan någon ytterligare nedladdningstid. I exempel 6.4 a placerar vi tre upplagor av samma bild efter varandra. Figur 6.4 b visar resultatet. Om du vill att bilderna ska hamna på separata rader måste du använda styckekoderna <br> eller <p> efter varje bild. Bilder beter sig på den punkten inte annorlunda än text.

Exempel 6.4 a
Här visas samma bild tre gånger...

```
<img src="bilder/storbild.gif" />
<img src="bilder/storbild.gif" />
<img src="bilder/storbild.gif" />
```

Figur 6.4 b
... och så här blir resultatet.

Figur 6.4 c (lilla bilden)
Netscape kan ibland skapa mellanrum mellan bilderna. Det är en ökänd bugg som vållat HTML-kodare över hela världen stor irritation.



I vissa webläsare, bland annat Nescape och Internet Explorer 4.0, kan det hända att bilderna skiljs åt av mellanrum (figur 6.4 c). Detta är ett problem som vållat HTML-kodare över hela världen timmar av frustrerat felsökande. Mellanrummen uppstår därför att webläsaren översätter radbrytningarna i själva HTML-dokumentet till mellanslag. Mellanrummen för-

svinner om du skriver hela koden på samma rad:

```
<img src="bilder/storbild.gif" /><img src="bilder/storbild.gif"
/><img src="bilder/storbild.gif" />
```

Enligt alla HTML-regler ska detta sätt att skriva HTML-kod ge exakt samma resultat som det i exempel 6.4 a, men så är alltså inte alltid fallet.

Att fördjupa sig i programmissar kan tyckas vara överkurs, men buggen med de mystiska mellanrummen är faktiskt något de flesta som gör hemsidor förr eller senare får problem med.

Om du använder ovanstående knep kommer din sida att se bra ut oavsett vilken webläsare dina besökare använder.

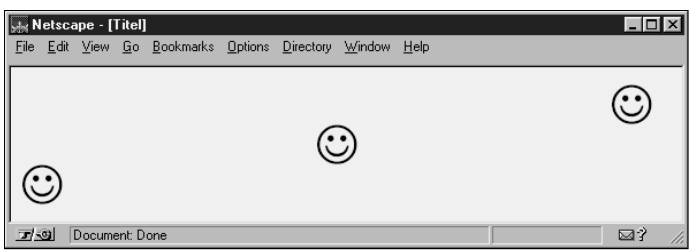
Att justera bilderna

Precis som med text finns det möjlighet att styra bildernas justering på sidan. Enklaste sättet är att använda samma koder som du använder för att justera text. Om du placerar före bilden och efter kommer den att bli centrerad. En annan variant är att använda <div align="*left", "center"* eller "*right*">. Hur det fungerar tillsammans med en bild syns i exempel 6.5 a och figur 6.5 b.

Exempel 6.5 a
Justeringskoden fungerar lika bra med bilder som med text.

```
<div align="right"><img src="bilder/minibild.gif" /></div>
<div align="center"><img src="bilder/minibild.gif" /></div>
<div align="left"><img src="bilder/minibild.gif" /></div>
```

Figur 6.5 b
Resultatet av kodningen i exempel 6.5 a.



Bildens placering i förhållande till text

När text och bild ska blandas på en sida räcker ofta inte och för att justera bilden. För det ändamålet finns ett särskilt attribut till <img />: align="*top", "middle", "bottom", "left"* och "*right"*. De första tre align-varianterna (*top, middle* och *bottom*) används när en bild ska sprängas in i en textrad, och justeringen styr bildens vertikala placering i förhållande till texten. innebär till exempel att bilden justeras efter textens mittpunkt. Exempel 6.6 a visar hur alla de tre justeringarna används och figur 6.6 b visar resultatet.

Exempel 6.6 a
Attributet align påverkar tillsammans med <img> bildens placering i förhållande till en textrad. Resultatet av kodningen visas i figur 6.6 b.

```
<img src="bilder/mellbild.gif" align="top" />Bilden justeras med
textens överkant (align="top")<p>
<img src="bilder/mellbild.gif" align="middle" />Bilden justeras
med textens mittlinje (align="middle")</p><p>
<img src="bilder/mellbild.gif" align="bottom" />Bilden justeras
med textens nederkant (align="bottom")
```

Figur 6.6 b
Resultatet av kodningen i exempel 6.6 a.

Flytande bilder

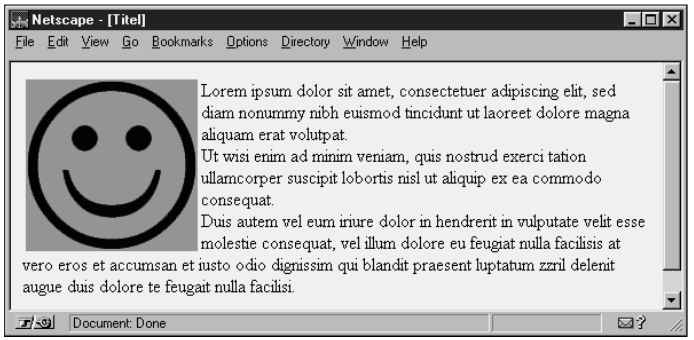
Align="*left"* och "*right"* har två effekter: dels visas bilderna till vänster respektive höger på sidan, dels hamnar efterföljande text längs med bildernas högra respektive vänstra sida. Detta kallas att bilden flyter.

Koden i exempel 6.7 a visar hur align="left" får texten att flyta runt bildens högersida och underkant. Figur 6.7 b visar resultatet. Prova gärna att istället skriva align="right". Bilden kommer då att hamna till höger, och texten kommer att flyta till vänster.

Exempel 6.7 a
Attributet align="left" får bilden att hamna till vänster, och den efterföljande texten att flyta runt bildens högersida.

```
<img src="bilder/storbild.gif" align="left" />
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.<br />
Ut wisi enim ad minim veniam (osv)
```

Figur 6.7 b
Resultatet av kodningen i exempel 6.7 a.



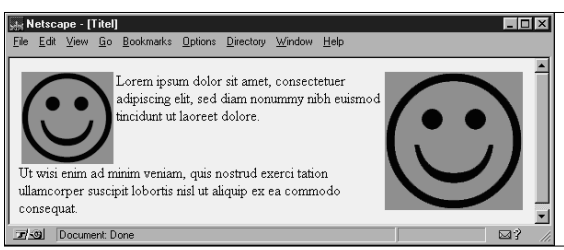
Att styra flytande text

I exempel 6.7 a och figur 6.7 b hamnar så mycket av texten som får plats på bildens högersida, medan resten fortsätter på raden under bilden. Det går också bra att styra hur stor del av texten som ska flyta vid sidan om bilden. Detta åstadkoms genom att koden för radbrytning, <br />, kompletteras med attributet clear=”*left”, “right”* eller “*all”*. <br clear=”left” /> innebär att textraden bryts och att nästa textrad dyker upp först på den rad vars vänstermarginal inte är upptagen av en bild. <br clear=”right” /> gör samma sak men letar efter en rad som inte innehåller någon högerställd bild. <br clear=”all” /> hoppar ned till en rad som inte innehåller några bilder på vare sig höger eller vänster sida. I exempel 6.8 a används <br clear=”left” /> för att få textens andra stycke att hamna under den vänstra, mindre bilden. Figur 6.8 b visar resultatet. Lägg märke till att den högra bilden sträcker sig längre ned än den vänstra, men att clear=”left” bara bryr sig om den vänstra bilden.

Exempel 6.8 a
Texten som följer efter <br clear="left" /> hamnar på första raden som inte innehåller någon vänsterställd bild.

```
<img src="bilder/mellbild.gif" align="left" />
<img src="bilder/storbild.gif" align="right" />

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh.<br clear="left" />Ut wisi enim ad (osv)
```

Figur 6.8 b
Resultatet av kodningen i exempel 6.8 a.



<br clear=”right” /> hade i exemplet ovan fått texten att hamna under högra bilden. Alternativet <br clear=”all” /> är bra att använda om man inte vet vilken av bilderna som är störst, men under alla omständigheter vill att texten ska hamna på en rad som är helt fri från bilder. I exemplet ovan skulle resultatet av <br clear=”all” /> blivit detsamma som om <br clear=”right” /> använts.

Luft runt bilder

När man använder flytande bilder omgivna av text är det ofta snyggt att ha lite luft mellan bilderna och texten. Två attribut till <img /> används för detta: hspace=”värde” (efter engelskans horizontal space, horisontellt utrymme) och vspace=”värde” (efter engelskans vertical space, vertikalt utrymme). I kodexempel 6.9 a har bilden fått ett 50 pixlar brett tomrum på sin högra och vänstra sida med hjälp av attributet hspace=”50”. Utrymmet över och under bilden är satt till fem pixlar med hjälp av attributet vspace=”5”. Figur 6.9 b visar resultatet.

vspace=”*värde”* och hspece=”*värde”* kan naturligtvis även användas för att skapa luft mellan olika bilder.

Exempel 6.9 a
vspace och hspace skapar luft vertikalt och horisontellt kring en bild.

```
<img src="bilder/storbild.gif" align="left"
vspace="5" hspece="50" />
Lorem ipsum dolor sit amet, consectetuer (osv)
```

Figur 6.9 b
Resultatet av kodningen i exempel 6.9 a.

Bilder som länkar

Att göra en bild till en klickbar länk är lätt. Skapa länken precis som vanligt (mer om detta i kapitel 4) men byt ut den klickbara texten mot en bild. Så här kan det se ut:

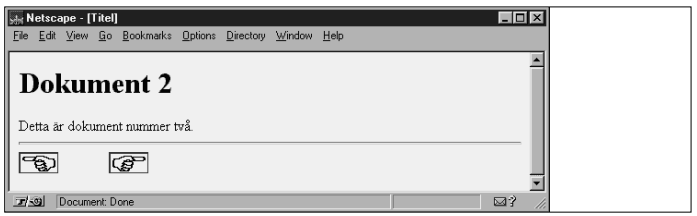
<**img src="sökväg och namn på bilden" />**

Bilden får automatiskt en blå ram för att skilja den från andra, icke klickbara bilder. Exempel 6.10 a visar hur bilderna ”peka_v.gif” och ”peka_h.gif” gjorts klickbara. Resultatet syns i figur 6.10 b; två pekande händer med vilka användaren enkelt kan klicka sig fram och tillbaka i ett större system av HTML-dokument. Lägg märke till att andra bilden, ”peka_h.gif”, fått en horisontell marginal på 50 pixlar genom attributet HSPACE=”50”. På så sätt hamnar den ett stycke ifrån den vänstra bilden.

Exempel 6.10 a
Om en bild placeras mellan länkkoderna och blir den en klickbar länk.

```
<h1>Dokument 2</h1>
Detta är dokument nummer två.
<hr>
<a href="dok1.html"><img src="bilder/peka_v.gif" /></a>
<a href="dok3.html"><img src="bilder/peka_h.gif"
hspace="50" /></a>
```

Figur 6.10 b
Resultatet av kodningen i exempel 6.10 a.



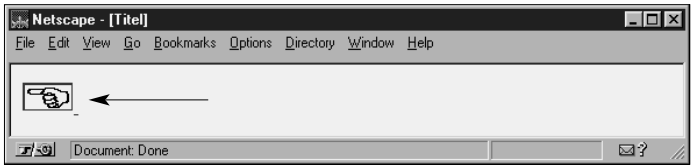
Undvik fula streck

Vi har tidigare i det här kapitlet nämnt ett programfel i Netscape som orsakar mellanslag mellan bilder som egentligen skulle ligga omedelbart intill varandra. Samma bugg ställer till problem när man gör en bild klickbar. Problemet uppstår om du i HTML-dokumentet byter rad före avslutningskoden . Netscape översätter då radbrytningen till ett mellanslag, och eftersom länktext ju är understruken blir resultatet ett understruket mellanslag direkt efter bilden. Exempel 6.11 a visar en länk där ligger på en separat rad i HTML-dokumentet. Figur 6.11 b visar det icke önskvärda streck som då dyker upp direkt efter den klickbara bilden. Om du istället flyttar upp en rad försvinner strecket.

Exempel 6.11 a
Om HTML-koden innehåller en radbrytning före ...

```
<a href="dok1.html"><img src="bilder/peka_v.gif" />
</a>
```
ny rad

Figur 6.11 b
... ger Netscape den klickbara bilden ett fult streck.



Att ändra ramens tjocklek

Det är ofta snyggt att ge bilden en ram. Det görs med attributet border=”*värde*”, som används tillsammans med <img>. Ramen kan vara hur tjock som helst, men en tunn ram är oftast snyggast.

Om ramen ska vara en pixel tjock ser koden ut så här:

```
<img src="bildens namn och sökväg" border="1" />
```

Tyvärr har Netscape och Internet Explorer länge tolkat attributet border olika. Internet Explorer 1.0 till och med 3.0 visar bara ramar på bilder som är klickbara. Andra bilder får istället en osynlig ram med den bredd som angivits med border-attributet. Alla versioner av Netscape samt Internet Explorer 4.0 ger synliga ramar till alla bilder som försetts med border-attribut. För att bilder med ram inte ska förväxlas med klickbara länkar får alla klickbara bilder blå ram istället för svart.

Koden i exempel 6.12 a visar en klickbar och en vanlig bild. Båda har en fyra pixlar tjock ram. Figur 6.12 b visar resultatet i Netscape. Den övre bilden har blå ram och är klickbar. Den undre har fått en svart ram. Figur 6.12 c visar hur Internet Explorer 3.0 tolkar koden. Nu har bara den klickbara bilden fått synlig ram medan den undre bildens ram är osynlig.

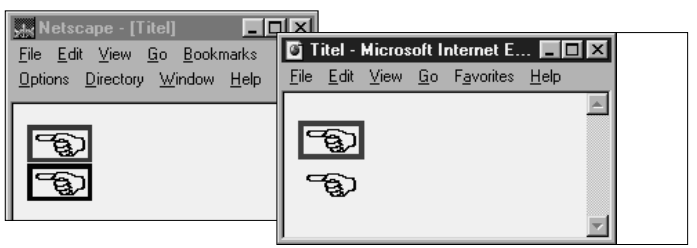
Om du inte vill ha någon ram alls trots att en bild är klickbar är det bara att använda attributet border=”0”. Här måste man dock vara försiktig. Nätsurfare är vana vid att klickbara bilder markeras med ram. Om ramarna försvinner finns det alltså risk för att dina besökare missar länkarna.

Exempel 6.12 a
Med attributet border kan ramstorleken på bilder ändras.

```
<a href="dok1.html"><img src="bilder/peka_v.gif"
border="4"></a><br />
<img src="bilder/peka_v.gif" border="4" />
```

Figur 6.12 b (vänster)
I Netscape kan alla bilder få en ram. Ramarna på icke klickbara bilder blir då svarta.

Figur 6.12 c (höger)
Internet Explorer 3.0 ignorerar border-attributet om bilden inte är klickbar. IE 4.0 fungerar dock som Netscape.



Viktigt.
Ange alltid bildens höjd och bredd med attributen heigh och width.
Det kan dramatiskt minska din hemsidas laddningstid.

Bildstorlekar

Med attributen height och width (höjd och bredd på engelska) talar du om för webläsaren hur hög och bred en bild är mätt i pixlar. Så här används koderna om bilden är 150 pixlar hög och 150 pixlar bred:

```
<img src="bilder/storbild.gif" height="150" width="150" />
```

Det är inte nödvändigt att ange height och width, men det finns ett mycket starkt skäl för att alltid göra det: det påskyndar laddningen av din hemsida. Om webläsaren vet bildernas mått kan den omedelbart efter att HTML-dokumentet anlänt reservera utrymme för bilderna, och därefter visa texten där den ska vara på skärmen. Bilderna syns då som tomma ramar som successivt fylls allteftersom bilderna laddas ned. Väntetiden kan besökaren använda till att läsa sidans texter.

Om bildernas mått inte anges måste webläsaren vänta med att visa texten till dess att samtliga bilder laddats ned och alla mått kontrollerats. Den kan nämligen inte veta exakt var på sidan texten ska visas utan att veta hur stort utrymme bilderna kommer att ta i anspråk. Om en eller flera bilder är stora och tidskrävande kan resultatet bli att användaren tvingas sitta och titta på en tom hemsida i flera minuter. Eftersom nätsurfare är ett otåligt släkte är risken då överhängande att du förlorar besökare som inte orkar vänta.

Om du inte vet hur stor en bild är kan du öppna den i nästan vilket grafikprogram som helst och där ta reda på måtten.

Att ändra storlek på bilder

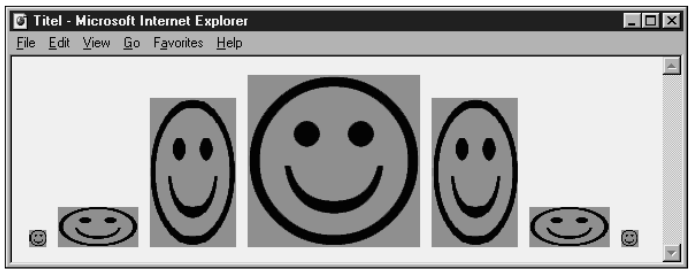
En rolig och användbar finess med koderna width och height är att de kan få webläsaren att förstora eller förminska en bild. Detta är mycket praktiskt om du till exempel vill ha med en återkommande symbol eller logotype på dina hemsidor. När den väl en gång laddats ned kan den dyka upp i flera olika storlekar utan extra nedladdningstid.

Exempel 6.13 a visar hur en bild som ursprungligen är 150 gånger 150 pixlar stor används flera gånger i olika storlekar. Lägg märke till att inte ens bildens proportioner måste stämma med originalet. Webläsaren drar ut eller pressar ihop bilden så att den stämmer med de mått du angivit.

Exempel 6.13 a
En bild visas i flera upplagor, med olika dimensioner och storlekar.

```
<img hspace="5" src="bilder/storbild.gif"
height="15" width="15" />
<img hspace="5" src="bilder/storbild.gif"
height="35" width="70" />
<img hspace="5" src="bilder/storbild.gif"
height="130" width="75" />
<img hspace="5" src="bilder/storbild.gif"
height="150" width="150" />
<img hspace="5" src="bilder/storbild.gif"
height="130" width="75" />
<img hspace="5" src="bilder/storbild.gif"
height="35" width="70" />
<img hspace="5" src="bilder/storbild.gif"
height="15" width="15" />
```

Figur 6.13 b
Resultatet av kodningen i exempel 6.13 a.



Viktigt.
Allt fler blinda surfar på www med hjälp av maskiner som översätter texten på hemsidorna till tal eller punktskrift. Om du använder bilder som klickbara länkar bör du därför alltid som alternativtext ange vart länken leder. Annars stänger du ute dem som inte kan se.
Jäktade surfare som stängt av bildvisningen av hastighetsskäl kommer också att gilla dig.

Alternativtexter

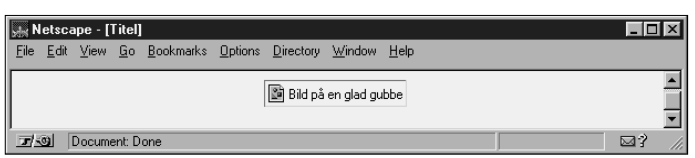
Om bildvisningen i webläsaren är avstängd markeras platsen där bilderna skulle ha visats med en liten symbol. Med hjälp av attributet alt="*alternativ text*" (efter engelskans *alternative*) kan du bistå dina besökare med mer information. ALT visar en text där bilden skulle ha visats. Detta är särskilt viktigt om bilden är en länk. Internet Explorer visar dessutom ALT-texten i en liten ruta när muspekaren förs över bilden, även då bildvisningen är påslagen.

Koden i exempel 6.14 a visar hur ALT används. Lägg märke till att texten måste stå inom citationstecken, eftersom den innehåller mellanslag. Figur 6.14 b visar resultatet i Netscape då bildvisningen är avstängd.

Exempel 6.14 a
Bilden får en alternativtext med hjälp av attributet ALT.

```
<center>
<img src="bilder/storbild.gif" alt="Bild på en glad gubbe" />
</center>
```

Figur 6.14 b
När bildvisningen är avstängd syns alternativtexten tillsammans med en bildsymbol.



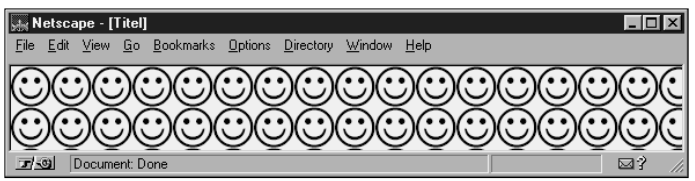
Bakgrundsbilder

För att förse en hemsida med bakgrundsbild ska vi för första gången använda ett attribut till <body>-koden. Attributet heter background=”*bakgrundsbild”,* och både Gif- och Jpeg-bilder kan användas. Eftersom webläsaren automatiskt lägger flera kopior av bilden bredvid varandra tills fönstret är fullt, kan en liten snabbladdad bakgrundsbild med fördel användas.

I exempel 6.15 a har vi gjort en liten bild på en glad gubbe till bakgrundsbild. Lägg märke till att det inte går att ange width, height eller några andra attribut till en bakgrundsbild, bara bildens sökväg och namn får finnas med. Figur 6.15 b visar resultatet.

Exempel 6.15 a
Bakgrundsbilden anges i <body>-koden.

```
<html>
<head><title>Titel</title>
<body background="bilder/minibild.gif">

</body>
</html>
```

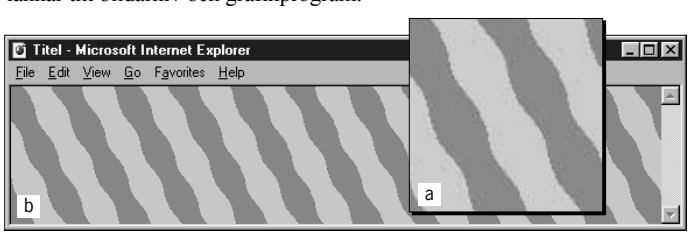


Figur 6.15 b
Resultatet. Webläsaren fyller automatiskt upp hela fönstret med bilden som angivits som bakgrundsbild.

På WWW finns många arkiv där du kan ladda ned gratis bilder särskilt skapade för att användas som bakgrundsbilder. Ofta är de liksom bilden i figur 6.16 a konstruerade för att fylla fönstret utan skarvar (figur 6.16 b).

Det är ganska vanligt att webdesigners ger sina sidor en färgad vänster- eller högermarginal med hjälp av en bakgrundsbild.

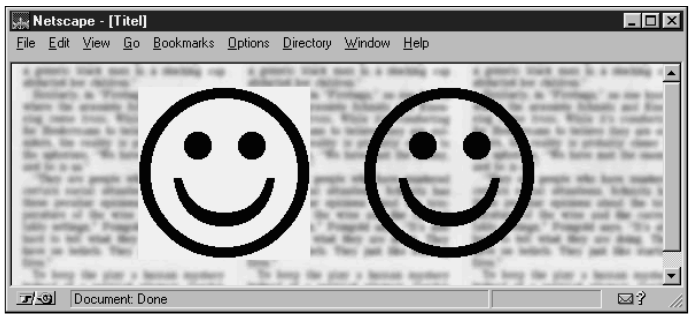
Hur detta går till går vi igenom på bokens hemsida. Gå till adress http://www.etc.se/nyckeln/ och välj Kapitel 6 i menyn. Där hittar du också länkar till bildarkiv och grafikprogram.



Figur 6.16 a och b
En bild som särskilt skapats för att vara bakgrundsbild (a) fungerar som ett pussel med en enda pusselbit när den duplicerats för att fylla ett helt fönster (b).

Transparenta Gif-bilder

När en bild ska visas på en sida med bakgrundsbild är det ofta mycket användbart att göra delar av bilden genomskinliga, *transparenta.* Bilden till vänster i figur 6.17 har vanlig, ogenomskinlig bakgrund, medan bakgrunden i bilden till höger är transparent.



Figur 6.17
Två olika typer av Gif-bilder. Bilden till höger är *transparent,* vilket innebär att en del av bilden blir genomskinlig då bilden visas på hemsidan. Detta är särskilt användbart på hemsidor som, liksom denna, har bakgrundsbild.

För att en bild ska bli transparent krävs att den sparas i ett särskilt Gif-format som heter GIF89a. Inte alla grafikprogram klarar detta, men med den takt World Wide Web växer lär funktionen snart bli standard.

Shareware.
Shareware-systemet bygger på att du som användare får prova ett program under en viss tid, ofta 30 dagar. Om du sedan vill fortsätta att använda programmet måste du betala för det. Detta kallas ofta att *registrera* programmet hos tillverkaren.

PC-användare

För PC-användare finns ett utmärkt **shareware**-program som heter LView som klarar att spara bilder som GIF89a. Programmet finns för gratis nedladdning på adress **http://www.lview.com**.

Version 4.0 och nyare versioner av det professionella (och därför dyra!) grafikprogrammet Photoshop är särskilt anpassat för produktion av grafik till websidor och klarar därför också transparenta Gif- och png-bilder.

Mac-användare

För Macintosh finns shareware-programmet Transparency. Det hittar du tillsammans med en rad andra program på filarkivet Tucows sida för grafikprogram: **http://www10.torget.se/tucows/mac/imgeditmac.html**

Precis som i PC-versionen klarar Photoshop 4.0 för Macintosh transparenta Gif-bilder.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Samtliga nedanstående koder fungerar med Netscape Navigator 2.0-4.0 (eller senare) och Internet Explorer 2.0-4.0 (eller senare).

HTML-kod

Placerar en bild på hemsidan.

Attribut till <img />

align="*top*", *"middle"*, *"bottom"*, *"left"* eller *"right"*
Avgör hur bilden ska justeras i förhållande till omgivande text.

alt="text"
Skapar en alternativ text.

border="värde"
Talar om hur bred bildens ram ska vara. Fungerar bara med klickbara bilder i Internet Explorer version 1 t o m 3.

height="värde"
Talar tillsammans med width om hur stor bilden ska vara, mätt i pixlar.

hspace="värde"
Talar om hur stor marginal bilden ska ha på sin högra sida.

src="sökväg och filnamn"
Talar om sökvägen till och namnet på bilden som ska visas. Gif, png och jpeg-bilder kan användas.

vspace="värde"
Talar om hur stor marginal bilden ska ha vertikalt.

width="värde"
Se height ovan.

<br clear="*left*", *"right"* eller *"all"* />

Används för att få en text att hamna på en rad som är fri från flytande bilder.

Attribut till <body>

background="*sökväg och filnamn"*
Bilden kopieras automatiskt i så många upplagor som behövs för att fylla webläsarens fönster.

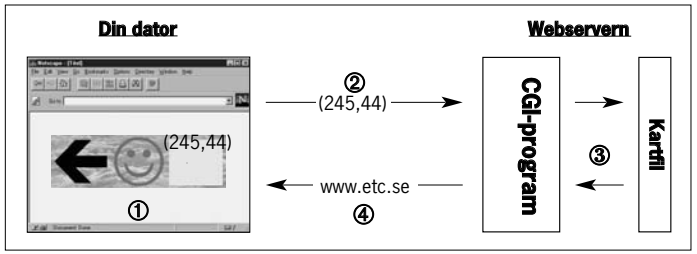
7 FLERA LÄNKAR I EN BILD

Bildkartor

På World Wide Web har du säkert ofta stött på klickbara bilder som leder till olika hemsidor beroende på var i bilden du klickar. Sådana bilder kallas *bildkartor* och i det här kapitlet ska vi gå igenom hur de konstrueras.

I förra kapitlet visade vi hur man gör en bild klickbar genom att placera den mellan länkkoderna <a> och . Med en sådan klickbar bild spelar det ingen roll var i bilden användaren klickar – bilden leder alltid till samma länk. Ibland kan det dock vara användbart att knyta olika delar av en bild till olika länkar. En sådan bild kallas för en *bildkarta* (*imagemap* på engelska). Namnet syftar på att en bildkarta består dels av en bild, dels av en *kartfil* som talar om vilka delar av bilden som är klickbara, och till vilka länkar de olika områdena ska leda.

Figur 7.1
En bildkarta är en bild som innehåller flera olika länkar. Nederst t h Netscapes variant där rutorna i bildens nederkant är klickbara. Överst t h MTVs hemsida där hela huvudmenyn är en bildkarta.
Stora bilden visar en mycket komplex bildkarta där varje ruta och cirkel är en egen länk.



Figur 7.2
För att en serverbaserad bildkarta ska fungera krävs dels ett CGI-program på webservern, dels en särskild kartfil som talar om för programmet vilken länk varje punkt i bilden är kopplad till.

Två olika sorters bildkartor

Det finns två olika typer av bildkartor. Den som använts längst på World Wide Web kallas *serverbaserad bildkarta* (*server-side imagemap* på engelska) och kräver tre komponenter för att fungera:

- En bild som inkluderas i HTML-dokumentet med hjälp av <img>-koden. För att tala om att bilden är en serverbaserad bildkarta används attributet ismap.
- En *kartfil* som innehåller information om vilka områden i bilden som är klickbara, och vilka adresser dessa områden ska leda till. Kartfilen lagras som en separat textfil på webservern (din Internetleverantörs dator) och ska ha förlängningen ”.map” i filnamnet.
- Ett särskilt **CGI-program** för hantering av bildkartor. Om du är ansluten till en Internetleverantör måste CGI-programmet vara installerat på dennes webserver.

Figur 7.2 visar vad som händer då användaren klickar på en serverbaserad bildkarta:

① Webbläsaren tar reda på koordinaterna för den punkt där användaren klickat. Koordinaterna är i det här fallet 245,44.

② Webbläsaren skickar koordinaterna till CGI-programmet på webservern.

③ CGI-programmet jämför koordinaterna med kartfilen och får fram länkadressen till det område där användaren klickat. I det här fallet är adressen www.etc.se.

④ CGI-programmet skickar tillbaka den rätta adressen till webbläsaren, som därefter kan hoppa till www.etc.se.

? CGI-program. Ett CGI-program är ett datorprogram som finns på webservern. Genom att anropa ett sådant program kan webbläsaren få hjälp med uppgifter den normalt inte klarar. I kapitel 13 får du veta mer om CGI-program.

En modernare form av imagemap kallas *klientbaserad bildkarta* (*client-side imagemap* på engelska). Namnet syftar på att klienten, det vill säga webbläsaren, själv klarar av att ta reda på inom vilket område användaren klickat, och till vilken länk detta område är kopplat. Det krävs alltså inget CGI-program på webservern för att denna typ av bildkarta ska fungera. En kartfil krävs fortfarande, men istället för att läggas i en separat fil på webservern inkluderas den i HTML-dokumentet.

En klientbaserad bildkarta är betydligt enklare att använda än en serverbaserad, och fungerar med alla nyare webbläsare. Äldre webbläsare, som till exempel Netscape 1.1, klarar dock bara serverbaserade bildkartor, varför vi i det här kapitlet i detalj kommer att gå igenom hur båda modellerna används. Även om du planerar att bara använda klientbaserade bildkartor (vilket du har goda skäl till) är det bra om du också läser instruktionerna för den serverbaserade varianten. Det kommer att göra det lättare för dig att förstå hur en bildkarta generellt fungerar.

Serverbaserade bildkartor

Serverbaserade bildkartor var den första typ av bildkartor som användes på World Wide Web. Det innebär att de även fungerar med äldre modeller av webbläsare, vilket kan vara en fördel om du vill vara säker på att alla besökare ska kunna använda dina bildkartor.

Det finns dock flera nackdelar med serverbaserade bildkartor. Som tidigare sagts kräver de ett särskilt CGI-program för att fungera, vilket innebär att du inte kan testa din imagemap utan att koppla upp dig mot din Internetleverantör och ladda upp filer till webservern. Detta kan göra arbetet med din hemsida mer tidskrävande. Serverbaserade bildkartor innebär också mer väntetid för användaren än om du använder klientbaserade. Processen i figur 7.2 tar nämligen en stund att gå igenom, medan ett klick på en klientbaserad kartbild processas omedelbart (mer om detta senare).

Med detta sagt är det dags att titta på hur en serverbaserad bildkarta skapas. Processen sker i fyra steg:

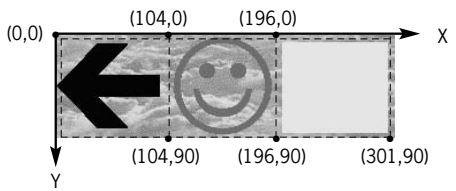
- Först bestäms koordinaterna för de delar av bilden som ska vara klickbara.
- Därefter ska dessa koordinater på rätt sätt noteras i *kartfilen*, som lagras som en separat fil i ditt hembibliotek på webservern. Det finns två olika sätt att skapa en kartfil. Vilken metod du ska använda beror på vilken typ av CGI-program din Internetleverantör installerat.
- Nu måste du ta reda på adressen till CGI-programmet på webservern, så att du kan anropa det från HTML-dokumentet. Detta sker enklast genom att helt enkelt fråga Internetleverantören.
- Slutligen inkluderas bilden i HTML-dokumentet, tillsammans med adressen till CGI-programmet. Kartfilen, HTML-dokumentet och bilden måste laddas upp till webservern innan din bildkarta kan testas.

Figur 7.3
Som exempel ska vi göra den här bilden till en serverbaserad bildkarta med tre klickbara områden.

Att bestämma koordinaterna

I det här exemplet ska vi göra bilden i figur 7.3 till en serverbaserad bildkarta innehållande tre olika länkar. Bilden heter ”klick.gif” och finns att hämta på bokens hemsida (www.etc.se/nyckeln/), men du kan naturligtvis också skapa en egen bild att experimentera med.

Först måste vi bestämma vilka områden som ska vara klickbara. För enkelhetens skull nöjer vi oss för tillfället med att dela upp bilden i tre rektangulära områden, som ska leda till varsin adress. I figur 7.4 är de tre områdena markerade med streckade linjer.

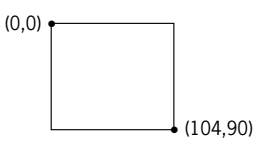


Figur 7.4
För att få fram koordinaterna tänker vi oss att bilden ligger i ett koordinatsystem där punkten 0,0 är överst till vänster.

För att ta fram koordinaterna för de tre rektanglarna tänker vi oss att bilden ligger i ett koordinatsystem där X-axeln ligger ovanför och Y-axeln till vänster om bilden (figur 7.4). Pixeln med koordinaterna 0,0 finns i bildens övre vänstra hörn, och ju längre nedåt och till höger vi rör oss i koordinatsystemet, desto högre blir koordinatvärdena.

Varje rektangel måste definieras med två koordinatpar: ett för övre vänstra hörnet och ett för nedre högra (figur 7.5). Eftersom X-värdet alltid skrivs först blir de två koordinatparen för den vänstra rektangeln 0,0 och 104,90. Nästa rektangel får värdena 104,0 och 196,90. Den högra rektangelns koordinater är 196,0 och 301,90.

Tips!
Det finns särskilda program som hjälper dig att ta fram koordinater i bilder. Ett lättanvänt program för både Windows och Mac heter ”MapEdit” och det kan laddas ned på adress **www.boutell.com/mapedit/**.



Figur 7.5
Varje rektangel måste definieras med två koordinatpar.

Två olika sorters kartfiler

Nu är det dags att skapa en *kartfil* där koordinaterna och länkadresserna ska skrivas in. Det finns två olika typer av kartfiler: CERN-modellen och NCSA-modellen. Vilken sort du ska använda beror på vilket CGI-program din Internetleverantör använder. Du måste alltså kontakta din Internetleverantör och fråga. Om leverantören har informationssidor på WWW finns där sannolikt uppgifter om vilken kartfil du ska använda.

Oavsett om du använder CERN- eller NCSA-modellen ska kartfilen sparas som ett vanligt textdokument. Filnamnet får inte innehålla mellanslag eller svenska tecken och ska avslutas med en punkt och förlängningen ”map” (till exempel ”min_karta.map”). Ett tips är att ge kartfilen samma namn som den bild kartan tillhör. Om bilden heter ”klick.gif” döps alltså kartfilen lämpligen till ”klick.map”.

Kartfil enligt CERN-metoden

Den här kartvarianten skapades av samma institution som uppfann hela World Wide Web: CERN (Conseil Européenne pour la Recherche Nucléaire i Genève, Schweiz). En kartfil enligt CERN-modellen ser ut som exempel 7.6 om vi använder koordinaterna i figur 7.4.

Exempel 7.6
En kartfil enligt CERN-metoden. Siffrorna till vänster är radnumreringar som inte ska ingå i kartfilen.

1. rect (0,0) (104,90) http://www.hotwired.com/index.html
2. rect (104,0) (196,90) http://www.etc.se/index.html
3. rect (196,0) (301,90) http://www.aftonbladet.se/start.html
4. default http://altavista.digital.com/index.html

Kartfilen rad för rad

1. Kartfilen inleds med ordet ”rect”. Det är en förkortning av *rectangle* och talar om att den första klickbara ytan som ska specificeras har rektangulär form. Därefter följer första koordinatparet, som omges av parenteser. Före andra koordinatparet måste det finnas ett mellanslag, annars fungerar inte kartfilen. Raden avslutas med en länkadress. Det är hit användaren kommer när han eller hon klickar på första rektangeln. Observera att hela sökvägen alltid måste anges. Relativa sökvägar fungerar inte. (Mer om relativa och absoluta sökvägar hittar du i kapitel 4.)

2–3. Eftersom vår bildkarta ska innehålla tre klickbara rektanglar upprepas rektangelspecifikationen på rad 2 och 3.

4. Sista raden innehåller inga koordinater, utan bara ordet ”default” (*frånvaro* på svenska) och en adress. Denna rad talar om vilken länk webbläsaren ska hoppa till om användaren klickar på en del av bilden som inte ligger inom ett definierat område. I figur 7.4 täcks visserligen hela bil-

Tips! Det gör inget om de områden du definierar i en bildkarta skulle överlappa varandra. Då är dock områdenas ordning i kartfilen viktig. När användaren klickar på en punkt som ingår i flera definierade områden blir det den länk som ligger överst i kartfilen som gäller.

dens yta in av de tre rektanglarna, varför default-adressen egentligen inte behövs. Det är dock bra att alltid ha med en default-adress, eftersom vissa CGI-program kan bli förvirrade om den saknas.

Kartfil enligt NCSA-metoden

Den andra modellen av kartfil utvecklades av NCSA (National Center for Supercomputing Applications vid University of Illinois, USA). Det var också här den första allmänt använda webbläsaren, Mosaic, utvecklades. En NCSA-karta fungerar i princip likadant som en CERN-karta, med den skillnaden att länkadressen placeras före koordinaterna, och att dessa inte ska omges av parenteser. Exempel 7.7 visar hur kartfilen till figur 7.4 ser ut enligt NCSA-modellen. Lägg märke till att koordinatparen fortfarande måste skiljas åt av ett mellanslag, och att default-adressen ser ut precis som i CERNs kartfil.

Exempel 7.7
En kartfil enligt NCSA-modellen. Lägg märke till att det *måste* finnas ett mellanslag mellan koordinatparen. Annars fungerar inte kartfilen.

```
rect http://www.hotwired.com/index.html 0,0 104,90
rect http://www.etc.se/index.html 104,0 196,90
rect http://www.aftonbladet.se/start.html 196,0 301,90
default http://altavista.digital.com/index.html
```

Att få in bildkartan i HTML-dokumentet

Nu har du en bild med tillhörande kartfil och det är dags att skriva själva HTML-dokumentet. För att göra detta måste vi introducera ett nytt attribut till <img />: ISMAP (från engelskans *is map*, ”är karta”). Som namnet antyder talar ISMAP om för webbläsaren att bilden som följer är en serverbaserad bildkarta.

Vi börjar med att placera bilden som ska bli en bildkarta på hemsidan. Om bilden ligger i katalogen ”bilder” blir koden så här:

```
<img src="bilder/klick.gif" ismap />
```

Tips! Om den som besöker din hemsida har stängt av bildvisningen kommer han eller hon inte att kunna använda dina bildkartor. Därför är det viktigt att ge användaren en chans att följa länkarna ändå. Detta åstadkommer du enkelt genom att upprepa länkarna i vanlig textform under den klickbara bilden.

Men vi är inte klara än. Nu måste bilden omges av länkkoderna <a> och . Länkadressen måste innehålla både adressen till CGI-programmet och adressen till kartfilen. Så här ska länken vara uppbyggd:

```
<a href="adressen till CGI-programmet/adressen till kartfilen"><img src="bilder/klick.gif" ismap /></a>
```

För att få reda på adressen till CGI-programmet är det åter dags att kontakta Internetleverantörens supportavdelning och fråga. En typisk adress kan vara ”/cgi-bin/imagemap” eller ”/htbin/htimage”.

Direkt efter adressen till CGI-programmet följer adressen till din kartfil. Hur adressen ser ut beror naturligtvis på var du valt att placera kartfilen. Det kan vara en god idé att lägga den i samma katalog som bilden. Om vi förutsätter att både bilden och kartfilen ligger i en katalog som heter ”bilder”, som i sin tur är en underkatalog till **”public_html”**, och att vårt användarnamn är ”Alice” kan den kompletta koden för vår serverbaserade bildkarta se ut som i exempel 7.7a.

Viktigt. "Public_html" är den katalog på webservern där alla dina hemsidor förvaras. I kapitel 12 får du veta mer om hur det går till att ladda upp filer till webservern.

Tänk på att du måste ladda upp bild, HTML-dokument och kartfil till webservern innan du kan testa om din nytillverkade serverbaserade bildkarta fungerar.

Exempel 7.7a
En serverbaserad bildkarta så som den ska se ut i HTML-dokumentet. Lägg märke till att länkadressen måste innehålla både adressen till CGI-programmet och kartfilen.

Länkkodens inledning **Sökväg till kartfilen**

```
<a href="/htbin/htimage/~alice/bilder/klick.map">
```

Sökväg till CGI-programmet

```
<img src="bilder/klick.gif" ismap /></a>
```

Bilden **Länkkodens avslutning**

Polygoner och cirklar

Förutom rektanglar går det bra att definiera polygoner och cirklar i en serverbaserad bildkarta. För att visa hur det går till ska vi återvända till bilden ”klick.gif” (figur 7.3), men denna gång göra polygonen och cirkeln (det vill säga pilen och ansiktet) till klickbara områden.

Den första formen, pilen, är en polygon och markeras med ordet **poly** i kartfilen. För att definiera en polygon måste koordinaterna för alla hörn, i det här fallet nio stycken, anges (figur 7.8). Du kan använda polygoner med upp till 100 hörn i en bildkarta.

Figur 7.8
För att definiera en polygon i en bildkarta måste koordinaterna för samtliga hörn anges.

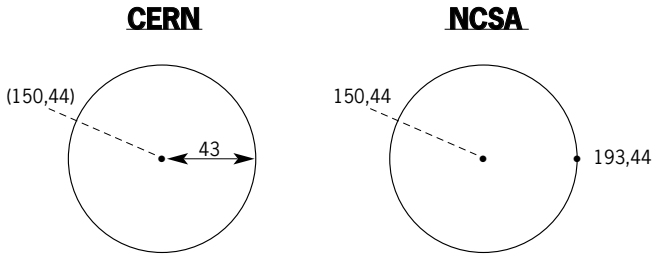
Koordinaterna förs in i kartfilen på samma sätt som om definitionen gällt en rektangel, men med fler koordinatpar. I CERN-modellen sätts koordinatparen alltså inom parentes och åtskiljs av mellanslag:

```
poly (41,7) (7,44) (41,80) (69,80) (44,53) (96,53) (96,35) (44,35)
(69,7) http://www.etc.se/index.html ny rad
```

Viktigt. Definitionen av ett klickbart område får inte innehålla radbrytningar. Markeringen "ny rad" visar var radbrytningar ska ske.

Enligt NCSA-metoden skulle samma figur se ut så här:

```
poly http://www.etc.se/index.html 41,7 7,44 41,80 69,80 44,53
96,53 96,35 44,35 69,7 ny rad
```



Figur 7.9
En cirkel definieras olika beroende på om din kartfil är av CERN- eller NCSA-modell. CERN-modellen kräver koordinaterna för cirkelns mittpunkt samt cirkelns radie mätt i pixlar. NCSA-modellen vill ha två koordinatpar: ett för cirkelns mittpunkt och ett för en punkt på cirkelns rand.
Under cirklarna visas den korrekta definitionen i respektive kartfilssystem.

Cirkeln markeras med **circle** i kartfilen och definieras olika beroende på om du använder NCSA eller CERN-kartan (se figur 7.9). I NCSA-kartan ska koordinaterna för två punkter anges: en som talar om var cirkelns mittpunkt ska vara belägen och en som definierar en punkt på cirkelns rand. Även i en CERN-karta ska cirkelns mittpunkt definieras, men sedan räcker det med att ange cirkelns radie mätt i pixlar.

Under cirklarna i figur 7.9 visas hur den korrekta definitionen ska se ut med CERN- respektive NCSA-systemet. Lägg märke till att siffran som anger cirkelns radie *inte* står inom parentes, som koordinaterna ju annars gör i en CERN-kartfil.

Viktigt.
I en kartfil måste varje definition av ett klickbart område ligga på en egen rad. Markeringen "ny rad" i exempel 7.10 och 7.11 visar var radbrytningar ska ske.

Exempel 7.10 (CERN) och 7.11 (NCSA) visar hur den kompletta kartfilen ser ut när pilen, cirkeln och kvadraten i figur 7.3 definierats som klickbara ytor. Lägg märke till att default-adressen nu fyller en viktig funktion, eftersom polygonen, cirkeln och kvadraten inte täcker hela bildens area. Om användaren klickar utanför de tre formerna hamnar han eller hon på AltaVistas söksida. Lämpligare är naturligtvis att låta default-länken gå till en särskild hjälpsida med information om var i bilden användaren ska klicka.

Exempel 7.10
En kartfil enligt CERN-metoden. Lägg märke till att cirkelns radie (här 43) inte ska stå inom parentes som koordinatparen.

```
poly (41,7) (7,44) (41,80) (69,80) (44,53) (96,53) (96,35) (44,35)
(69,7) http://www.hotwired.com/index.html ny rad
circle (150,44) 43 http://www.etc.se/index.html ny rad
rect (201,7) (293,84) http://www.aftonbladet.se/start.html ny rad
default http://altavista.digital.com/index.html ny rad
```

Exempel 7.11
Samma karta enligt NCSA-metoden.

```
poly http://www.hotwired.com/index.html 41,7 7,44 41,80 69,80
44,53 96,53 96,35 44,35 69,7 ny rad
circle http://www.etc.se/index.html 150,44 193,44 ny rad
rect http://www.aftonbladet.se/start.html 201,7 293,84 ny rad
default http://altavista.digital.com/index.html ny rad
```

Klientbaserade bildkartor

Den här typen av bildkarta är relativt ny på World Wide Web, och du kommer snart att märka att den har många fördelar framför den äldre, klientbaserade bildkartan. Här är de viktigaste:

- Hanteras helt och hållet av webbläsaren och kräver därför inget CGI-program på servern för att fungera.
- Är enklare att konstruera än serverbaserad bildkarta eftersom kartan inte sparas som separat fil utan inkluderas i HTML-dokumentet.
- Kan testas utan att först laddas upp till Internetleverantörens webserver.
- Klarar relativa sökvägar.
- Fungerar till skillnad från serverbaserade bildkartor tillsammans med rutor (*frames*). Mer om detta i kapitel 10.

Nackdelen med klientbaserade bildkartor är att de inte fungerar med riktigt gamla webbläsare, till exempel Netscape 1.1. Detta problem kommer du dock runt genom att alltid komplettera din bildkarta med vanliga textlänkar. Den besökare vars webbläsare inte klarar klientbaserade bildkartor kan då alltid använda textlänkarna istället. Om det inte är absolut nödvändigt att alla dina besökare får tillgång till klickbara bilder finns det därför goda skäl att bara använda klientbaserade bildkartor på din hemsida.

En klientbaserad bildkarta består av två delar:

- En bild som i vanlig ordning inkluderas i HTML-dokumentet med hjälp av <img />-koden. För att tala om att bilden är en klientbaserad bildkarta används dessutom attributet usemap="*#karta"*.
- En karta som inkluderas i HTML-dokumentet.

Kartan i HTML-dokumentet

Två nya HTML-koder används för att få in kartan i HTML-dokumentet. Här är båda två med alla sina attribut:

- <map> med avslutningskod </map> inleder och avslutar kartan. Attributet name="*kartnamn"* döper kartan.
- <area /> inleder definitionen av varje enskilt fält i figuren. Attributet shape="*circle", "poly", "rect"* eller "*default"*, talar om vilken form eller funktion ett område har. Attributet coords="*koordinater"* anger områdets koordinater. Attributet href="*adress"* anger områdets länkadress.

Figur 7.12
Som exempel ska vi göra pilen, cirkeln och kvadraten till en klientbaserad bildkarta.

Nu ska vi göra en klientbaserad bildkarta av figur 7.12, där pilen, cirkeln och rektangeln ska vara klickbara.

Vi börjar med att ta fram figurernas koordinater. Det sker på samma sätt som när vi gjorde en serverbaserad bildkarta (se sidan 74).

Nästa steg är att skriva in kartan i HTML-dokumentet. Öppna alltså ett nytt HTML-malldokument och knappa in koden i exempel 7.13 i dokumentets <BODY>-del. Nedan följer en genomgång av koden rad för rad.

Exempel 7.13
Kartdelen till en klientbaserad bildkarta av figur 7.12, så som den ska se ut i HTML-dokumentet. Lägg märke till att raden av koordinater sätts inom citationstecken.

```
1. <map name="klick">
2. <area shape="poly" coords="41,7, 7,44, 41,80, 69,80, 44,53,
   96,53, 96,35, 44,35, 69,7" href="dokument1.html" />
3. <area shape="circle" coords="150,44, 43"
   href="dokument2.html" />
4. <area shape="rect"  coords="201,7, 293,84"
   href="dokument3.html" />
5. <area shape="default" href="dokument4.html" />
6. </map>
```

Koden rad för rad

1. Första raden inleds med kartkoden <map>. Den talar om för webbläsaren att en karta nu inleds. name="*klick"* ger kartan namnet "klick".

2. Andra raden inleds med <area>, vilket talar om att ett klickbart område nu ska definieras. Därefter följer attributet shape (*form* på svenska). De tre formerna som kan användas känner du igen från kartfiler skrivna enligt både CERN och NCSA-modellen: *circle, rect* och *poly* (se avsnittet om serverbaserade bildkartor tidigare i kapitlet). I det här fallet är det en polygon som ska definieras, vilket anges med shape="*poly"*.

Attributet coords (efter engelskans *coordinates*, koordinater) inleder uppräkningen av polygonens nio koordinatpar. Lägg märke till att **samtliga** koordinater ska skiljas åt med kommatecken. Det går bra att, som i exemplet, dessutom skilja koordinatparen åt med hjälp av mellanslag, men då är det mycket viktigt att hela raden av koordinater sätts inom citationstecken.

Raden avslutas med attributet href=dokument1.html. Det är alltså till "dokument1.html" webbläsaren kommer att hoppa då besökaren klickar på polygonen. Lägg märket till att sökvägen är relativ (den innehåller ju bara namnet på HTML-dokumentet). Klientbaserade bildkartor är den enda typ av bildkartor som klarar både relativa och absoluta sökvägar.

3. På den här raden definieras cirkeln med hjälp av shape="circle". Här måste CERN-metodens sätt att beskriva en cirkel användas. De första två koordinaterna anger alltså cirkelns mittpunkt, medan den tredje siffran talar om cirkelns radie. Värdena separeras med kommatecken. Om mellanslag läggs in måste alla koordinater omges av citationstecken.

4. Slutligen definieras rektangeln med attributet shape="rect".

5. För att ange en default-adress används attributet shape="default". Den här adressen kommer användaren till om han eller hon klickar utanför polygonen, cirkeln eller kvadraten.

6. Slutligen avslutas hela kartan med </map>.

Länklösa områden med nohref

Om du inte vill att det ska hända någonting alls då användaren klickar inom ett särskilt område, kan du byta ut href="*länkadress"* mot attributet nohref. Om vi i exemplet ovan inte ville att rektangeln skulle leda till någon länk skulle vi alltså ha skrivit:

```
<area shape="rect"  coords="201,7, 293,84" nohref />
```

Att få in bilden i HTML-dokumentet

Om du knappat in koden i exempel 7.13 har du nu ett HTML-dokument med en karta i. Återstår att lägga in bilden som ska bli klickbar. Det gör vi med hjälp av ett nytt attribut till <img />: usemap="*#kartnamn"*.

Skriv alltså in kodraden i exempel 7.14 före eller efter (det spelar ingen roll vilket) kartan i HTML-dokumentet. Vi utgår ifrån att bilden ligger i katalogen "bilder".

Exempel 7.14
Så här ser raden ut som placerar den klientbaserade bilden på sidan, och kopplar den till rätt karta.

```
<img src="bilder/klick.gif" usemap="#klick" />
```

Lägg märke till att attributet usemap="#klick" knyter bilden till kartan, som ju fick namnet "klick" i raden <map name="klick">.

Och så var det klart! Spara dokumentet och ladda in det i din webbläsare. Om du skrivit rätt kommer din klientbaserade bildkarta nu att fungera. En förutsättning är naturligtvis att de dokument som länkarna leder till verkligen existerar. Om bildkartan ändå inte fungerar bör du noga kontrollera att alla citationstecken i kartan verkligen sitter där de ska. Ett enda citationstecken för mycket räcker för att göra en karta oanvändbar.

Separata dokument för kartorna

I vårt exempel ligger kodraden som anropar bilden ”klick.gif” () och kartan i samma HTML-dokument. Det är oftast mest praktiskt. Men det går också att lägga kartan i ett separat dokument. Då måste raden som anropar bilden utökas så att den innehåller både sökvägen till dokumentet och namnet på kartan.

Om vi till exempel hade placerat kartan ”klick” (exempel 7.14) i ett särskilt HTML-dokument som heter ”kartor.html” ser raden som anropar bilden ut så här:

```
<img src=”bilder/klick.gif” usemap=”kartor.html#klick” />
```

Lägg märke till att adressen efter usemap nu innehåller sökvägen till HTML-dokumentet med kartan (kartor.html) direkt följt av ett brädgårdstecken (#) och namnet på kartan. Om vi istället använt absolut sökväg hade kodraden sett ut så här:

```
<img src=”bilder/klick.gif”
usemap=”http://www.etc.se/kartor.html#klick” />
```

Falska bildkartor

När ett antal vanliga klickbara bilder ligger tätt intill varandra och saknar ramar går de inte att skilja från en bildkarta. Detta kallas ibland för en *falsk bildkarta* och är ett flexibelt och praktiskt alternativ till riktiga bildkartor.

För att göra en falsk bildkarta av bilden ”klick.gif” (stora bilden i figur 7.15) klipper man helt enkelt sönder den i tre delar: ”klick1.gif”, ”klick2.gif” och ”klick3.gif” (de små bilderna i figur 7.15). Nästa steg är att göra de tre bilderna klickbara med hjälp av länkkoderna <a> och . Ramen som alla klickbara bilder automatiskt får tas bort med attributet border=”0”. Resultatet blir att bilderna flyter ihop och inte längre går att skilja från en riktig bildkarta.

Figur 7.15
Genom att klippa sönder den större bilden till tre små kan man skapa en så kallad *falsk bildkarta* som fungerar lika bra som en riktig bildkarta.

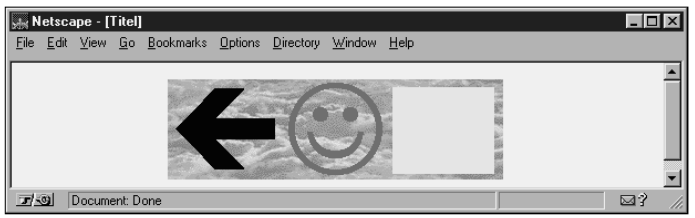
Exempel 7.16 a visar koden som sammanfogar de tre bilderna till en falsk bildkarta. Figur 7.16 b visar resultatet.

Tänk på att alla tre bilderna och avslutningskoderna </A> måste ligga på samma rad i HTML-dokumentet. Annars tolkar Netscape radbrytningarna som mellanslag, och du får mellanrum mellan bilderna.

Exempel 7.16 a
Genom att ta bort ramarna med border=”0” och lägga bilderna direkt intill varandra flyter de ihop och ser ut som en enda bild.

```
<a href=”dok1.html”><img src=”bilder/klick1.gif”
border=”0” /></a><a href=”dok2.html”><img
src=”bilder/klick2.gif” border=”0” /></a><a
href=”dok3.html”><img src=”bilder/klick3.gif”
border=”0” /></a>
```

Figur 7.16 b
Resultatet av koden i exempel 7.16 a.



Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Förkortningarna till höger anger med vilka webbläsare koderna och attributen fungerar.
ns = Netscape Navigator, ie = Internet Explorer. Siffrorna anger version.

KODER SOM SKAPAR BILDKARTOR

HTML-kod	**Webbläsare**
<img />	alla
Placerar en bild på hemsidan	
Attribut till <img />	
ismap	alla
Talar om att bilden är en serverbaserad bildkarta.	
src=”*sökväg och filnamn*”	alla
Talar om sökvägen till och namnet på bilden som ska visas. Gif-, png- och Jpeg-bilder kan användas.	
usemap=”*sökväg#kartnamn*”	alla
Anropar en klientbaserad bildkarta. Om sökvägen utelämnas och bara #*kartnamn* anges antas kartfilen finnas i samma HTML-dokument som <img />-koden.	

KODER SOM SKAPAR EN KARTA (klientbaserade bildkartor)

Här följer en sammanfattning av alla koder i kapitlet. Samtliga nedanstående koder fungerar med Netscape Navigator 2.0-4.0 (eller senare) och Internet Explorer 2.0-4.0 (eller senare).

HTML-kod

<map>…</map>
Startar och avslutar en karta.

Attribut till <map>
namne="*kartnamn*"
Ger kartan ett namn.
<area />
Definierar ett klickbart område.

Attribut till <area />
shape="*form*"
Anger områdets form. Kan vara *rect (*rektangel), *poly* (polygon), *circle* (cirkel) eller *default*.
coords="*koordinater*"
Anger områdets koordinater enligt följande:
Rektangel: x1,y1,x2,y2
Cirkel: *x,y,r* där *x,y* är cirkelns mittpunkt och *r* radien.
Polygon: x1,y1,x2,y2,x3,y3 osv
href="*sökväg*"
Anger vilken länk som ska vara kopplad till området.
nohref
Ersätter href="*sökväg*" och gör så att ingenting händer om användaren klickar i området.

SYNTAX FÖR KARTFILER (serverbaserade bildkartor)

Alla länkadresser måste utgöras av absoluta sökvägar.

CERN-modellen
rect (x1,y1) (x2,y2) *länkadress*
poly (x1,y1) (x2,y2) (x3,y3) osv *länkadress*
circle (x1,y1) r *länkadress*

NCSA-modellen
rect *länkadress* x1,y1 x2,y2
poly *länkadress* x1,y1 x2,y2 x3,y3 osv
circle *länkadress* x1,y1 x2,y2

8 STYR HEMSIDANS FÄRG:

Färgkoder

I början var WWW svartvitt. Sedan släppte Netscape sin webbläsare Netscape Navigator och det blev fritt fram att ändra allt från en hemsidas bakgrundsfärg till färgen på enskilda bokstäver. Idag är färgkoderna standard och i det här kapitlet får du lära dig att använda dem.

Hittills har vi bara använt ett attribut till <body>-koden: background=”*bakgrundsbild”*. Med detta attribut fick hela hemsidan en bakgrundsbild. De flesta koderna som ger färg åt hemsidan är, liksom background, attribut till <body>, och påverkar därför hemsidan i sin helhet. Här är alla attributen:

bgcolor=”#*färgkod”*

Bestämmer sidans bakgrundsfärg. Namnet kommer från engelskans *background color* (bakgrundsfärg). Om ingen bakgrundsfärg anges blir bakgrunden vit eller grå beroende på webbläsare.

text=”#*färgkod”*

Bestämmer textfärgen. Om ingen textfärg anges blir texten svart.

link=”#*färgkod”*

Bestämmer vilken färg länkarna ska ha (länk heter *link* på engelska). link-färgen påverkar dels all klickbar text, dels ramen på alla klickbara bilder. Om ingen länkfärg anges blir länkarna automatiskt blåfärgade.

vlink=”#*färgkod”*

Vlink har fått sitt namn efter engelskans *visited link* (besökt länk) och bestämmer färgen på de länkar som besökaren redan klickat på.

alink=”#*färgkod”*

Med alink (från engelskans *active link*, aktiverad länk) anger du slutligen vilket färg länken har i det ögonblick användaren klickar på den.

För att känna på färgkoderna kan du öppna ett nytt malldokument och knappa in koden i exempel 8.1 a. Spara sedan dokumentet och öppna det med din webbläsare. Resultatet kommer att bli en sida med illröd bakgrund, grön text och gulfärgad länktext (en svartvit version ser du i figur 8.1 b). De sex siffror eller bokstäver som kommer efter brädgårdstecknet (till exempel ”00FF00”) är färgkoder. Vi återkommer strax till hur dessa fungerar.

Exempel 8.1 a
Genom att lägga till attribut till <body>-koden påverkas färgen på hela hemsidan. Här gör vi bakgrunden illröd, texten grön och länkarna gula. En svartvit version av resultatet ser du i figur 8.1 b.

```
<html>
<head><title>Titel</title></head>
<body bgcolor="#FF0000" text="#00FF00" link="#FFFF00">

<font size="5">
Texten är grön och <a href="dummy.html">länken</a> är gul.
</font>

</body>
</html>
```

Figur 8.1 b
Resultatet av koden i exempel 8.1 a.

De sexsiffriga färgkoderna

Alla färgattribut avslutas med ett brädgårdstecken följt av en sexsiffrig färgkod. I exempel 8.1 a blev till exempel sidan illgrön genom attributet bgcolor=”#FF0000”. ”FF0000” är alltså färgkoden för illgrönt.

För att förstå hur färgkoderna fungerar måste vi titta på hur datorer gör för att visa färger.

Färgsystemet som används heter RGB och har fått sitt namn av initialerna i de tre färger som systemet bygger på: rött, grönt och blått. Alla färger i RGB-systemet bestäms av hur stor del rött, grönt respektive blått de innehåller. Värdet kan vara mellan 0 och 255. RGB-färger beskrivs därför med tre tal mellan 0 och 255, varav det första anger mängden rött, det andra mängden grönt och det sista mängden blått.

RGB-färgen 255,0,0 är exempelvis illröd, eftersom den innehåller en maximal dos rött (255) men inget grönt eller blått (0 och 0). På samma sätt är RGB-färgen 0,255,0 knallgrön och blandningen 237,113,200 blekt lila.

RGB-värdet för vitt är 255,255,255 och svart 0,0,0. Tre lika stora värden ger olika nyanser av grått. Värdena 20,20,20 ger alltså en ljusgrå nyans, medan 200,200,200 blir en mörkgrå.

Tips!
Om du använder en mörk bakgrundsbild och vit text kan det bli problem för besökare som stängt av bildvisningen. När bakgrundsbilden försvinner hamnar nämligen den vita texten på vit eller ljusgrå botten och blir oläslig. Lösningen är att alltid ange en bakgrundsfärg i samma ton som bakgrundsbilden. Om bilden försvinner kommer bakgrundsfärgen att bli synlig, vilket gör texten läslig.

I HTML-dokument är det färgens RGB-värde som ska anges. Men för att göra det hela lite mer komplicerat räcker det inte att skriva bgcolor=”#255,0,0” om man vill att sidan ska ha en röd bakgrundsfärg. De tre talen måste nämligen uttryckas *hexadecimalt*.

Hur det hexadecimala talsystemet fungerar kan du se i faktarutan nedan. Du behöver dock inte kunna räkna hexadecimalt för att få färg på din hemsida. Det räcker att du vet att ett hexadecimalt tal kan innehålla både sifror (0–9) och bokstäver (A–F). Det högsta talet i RGB-systemet, 255, blir FF uttryckt hexadecimalt, och rött uttrycks därför ”#FF0000” i HTML-dokumentet.

Att hitta rätt färger

Det finns några hjälpmedel för dig som inte är intresserad av att behöva sitta och räkna om RGB-värden till hexadecimala tal. Editorer som Dreamweaver och BBEdit visar helt enkelt en färgpalett där du enkelt kan klicka på de färger du vill ha på sidans länkar och text. Programmen genererar sedan automatiskt rätt <body>-kod.

Tips!
Tänk på att vissa färgkombinationer kan göra texten på din sida svårläst, särskilt för dem som kanske inte har en lika skarp monitor som du. Se alltså till att ha en någorlunda kontrast mellan texten och bakgrundsfärgen.

För dig som inte använder en HTML-editor finns samma hjälp att få på websidan Colormaker (adress: http://www.bagism.com/colormaker/). Här genereras dessutom en exempelsida där du snabbt kan se om färgkombinationerna som du valt är lyckade.

Ett hett tips är att inte sväva ut för mycket när du ger din sida färg. Svart bakgrund med rosa text och illgröna länkar är kanske roligt en stund, men ger också din sida ett amatörmässigt utseende. Om du inte särskilt är ute efter den effekten så använd mer harmoniska kombinationer.

Det hexadecimala talsystemet

Det talsystem vi vanligen använder består av tio siffror, 0–9. Det är därmed ett decimalt talsystem. Det hexadecimala talsystemet har istället 16 siffror. Eftersom vi inte har tecken för mer än tio siffror tar man i det hexadecimala talsystemet till bokstäver för att symbolisera de siffror som saknas. Att räkna till 15 enligt det decimala respektive det hexadecimala talsystemet går till så här:

Decimala systemet:	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Hexadecimala systemet:	0 1 2 3 4 5 6 7 8 9 A B C D E F

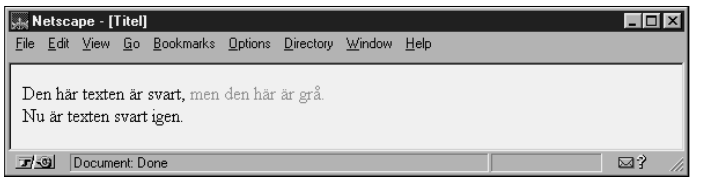
Ett hexadecimalt tal kan alltså bestå av både siffror och bokstäver (från A till F). Om du till exempel vill veta hur talet 202 uttrycks hexadecimalt kan du räkna så här: dela 202 med 16. Resultatet blir 12,625. Av tabellen ovan framgår att 12 blir C uttryckt hexadecimalt. Multiplicera nu det som blir kvar, 0,625, med 16. Resultatet blir 10, vilket blir A uttryckt hexadecimalt. 202 blir alltså CA uttryckt hexadecimalt.

Att ändra färg på delar av texten

Med text="*#färgkod"* bestämmer du vilken färg all text på sidan ska ha. Men det går också bra att ändra färg på vissa stycken, ord eller till och med bokstäver. Detta görs med koden och attributet color="*#färgkod"*. Exempel 8.2 a visar hur några ord mitt i en mening får en ny färg. Figur 8.2 b visar resultatet. Med hjälp av avslutningskoden återställs textfärgen till den du angivit med text="*#färgkod"* tillsammans med <body>.

Exempel 8.2 a
Med koden kan färgen på textavsnitt, ord eller till och med enskilda bokstäver ändras. återställer den ursprungliga färgen.

```
Den här texten är svart, <font color="#999999">men den här är grå.</font><br />
Nu är texten svart igen.
```



Figur 8.2 b
Resultatet av koden i exempel 8.2 a.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Förkortningarna till höger anger med vilka webbläsare koderna och attributen fungerar.
ns = Netscape Navigator, ie = Internet Explorer. Siffrorna anger version.

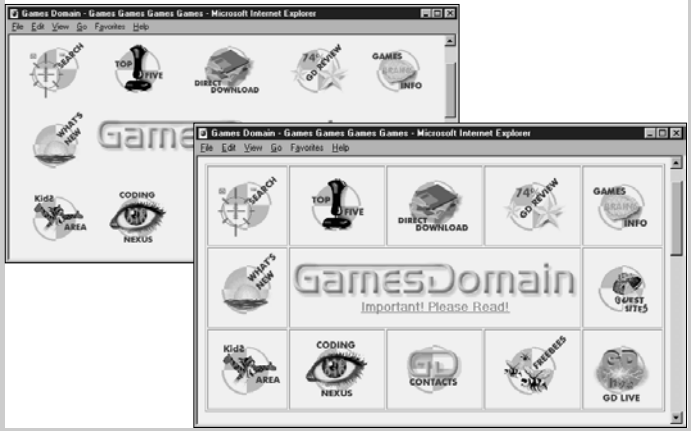
HTML-kod	Webbläsare
Attribut till <body>	
bgcolor="*#färgkod"*	alla
Ändrar bakgrundsfärg på hela hemsidan.	
link="*#färgkod"*	alla
Anger färgen för icke besökta länkar.	
text="*#färgkod"*	alla
Anger hemsidans textfärg.	
vlink="*#färgkod"*	alla
Anger färgen på besökta länkar.	
alink="*#färgkod"*	ie4·ns2-4
Anger färgen på aktiv länk (dvs vid klick).	
**Attribut till **	
color="*#färgkod"*	alla
Ändrar färg på efterföljande text. Originalfärgen återställs med .	

9 STYR DIN HEMSIDAS FORM:

Tabeller

Att det finns goda möjligheter att skapa tabeller i HTML beror på att språket främst skapades för publicering av forskningsrapporter. Snart visade det sig dock att tabeller faktiskt också kan användas som layoutverktyg. Idag är det svårt att hitta några mer ambitiösa hemsidor som inte utnyttjar tabeller för att styra placering av bilder och text. I det här kapitlet lär du dig alla knep som gör en tabell till mycket mer än bara en tabell.

De flesta nätsurfare tror sig nog känna igen en tabell när de ser en; det är ett rutsystem som används till att presentera statistik på ett överskådligt sätt. Vad inte lika många vet är att osynliga tabeller ofta används för att styra placeringen av bilder och text på en hemsida. I själva verket är det idag vanligare att en tabell används för layout än för att presentera statistik. Ett typexempel ser du i figur 9.1 och 9.2.



Figur 9.1
På spelcentralen Games Domains hemsida omges företagets logotype av klickbara bilder som placerats ut med jämna mellanrum.

Figur 9.2
I själva verket är hela sidan en tabell.
Det är mycket vanligt att hemsidesmakare på det här sättet styr placeringen av bild och text på sina sidor.

Att lära sig göra tabeller är alltså ett stort steg mot större kontroll av hemsidans utseende, och väl värt mödan även för den som inte planerar att presentera en enda siffra på sin hemsida.

Även om många nybörjare med visst fog tycker att tabeller är svårt, ska du inte låta dig avskräckas. Inlärningsprocessen påminner faktiskt om att lära sig cykla: Det kräver en del övning, men när man väl kan knepet är det väldigt lätt.

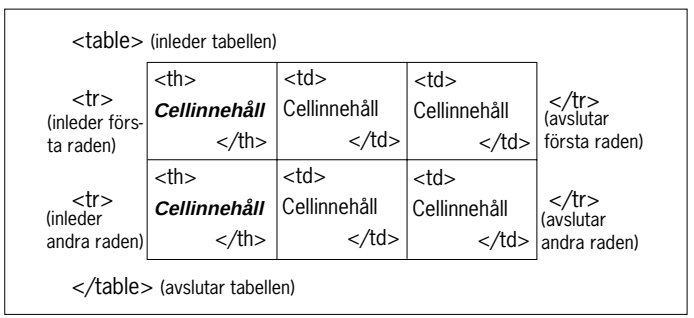
Kapitlet innehåller många exempel på hur olika typer av tabeller kodas. Ju fler du själv knappar in och experimenterar med, desto fortare kommer du att lära dig. Vissa kodningar är långa, och om du vill slippa skriva in dem för hand kan du hämta dem på bokens hemsida. Du når den på adress http://www.etc.se/nyckeln/. Där hittar du också samtliga bilder som finns med i exemplen.

Tabellens uppbyggnad

Rutorna i en tabell brukar kallas *celler,* och dessa är organiserade i *rader* och *kolumner.* Det finns två olika sorters celler: *rubrikceller* och *vanliga celler*. Rubrikcellerna brukar vara placerade överst eller till vänster i tabellen för att ge en rubrik åt varje individuell rad eller kolumn. I de vanliga cellerna placeras tabellens egentliga innehåll, vilket kan vara text, bilder eller till och med andra tabeller. Om en tabell ska användas till layout (som i figur 9.1 och 9.2) behövs naturligtvis inga rubrikceller.

Koderna som bygger tabeller

Det finns sex särskilda koder för att bygga tabeller i HTML. Figur 9.3 visar allihop. Kodernas placering visar deras funktion.



Figur 9.3
Koderna som skapar en tabell. Figuren ska läsas från vänster till höger. Kodernas placering visar vilken funktion de har i tabellen.

<table>...</table> börjar och avslutar hela tabellen.

<tr>...</tr> (från engelskans *table row*) inleder och avslutar varje rad i tabellen.

<td>...</td> (från engelskans *table data*) inleder och avslutar varje individuell cell i tabellen. Mellan dessa koder placeras tabellens innehåll.

<th>...</th> (från engelskans *table header*) inleder och avslutar varje individuell rubrikcell i tabellen. Fungerar precis som <td>, med den skillnaden att textinnehållet i en <th> centreras och sätts med fet stil. Det är inte nödvändigt att ha med någon <th> i tabellen.

De flesta koderna har dessutom attribut som vi återkommer till längre fram i kapitlet. Attributet border är dock så viktigt att vi tar det med en gång. Det används tillsammans med <table> och skapar en ram runt tabellen.

En tabell med ram inleds alltså så här:

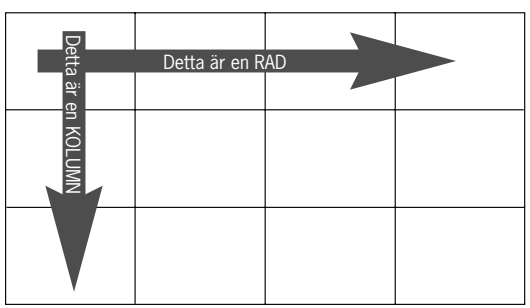
```
<table border>
```

Om tabellen inte fungerar

Vi ska strax börja koda tabeller, men först några ord om felsökning. Tabeller är mycket känsliga för felaktigheter i HTML-koden. Om du råkar ut för en tabell som inte ser frisk ut bör du först kontrollera att inga extra ”<” eller ”>” råkat smyga sig in. En snabb metod är att leta efter ”<<” och ”>>” med hjälp av texteditorns sökfunktion.

Om inte detta hjälper är chansen stor att du missat en avslutningskod. Kontrollera noga att alla <table>, <tr>, <td> och <th> också avslutas med </table>, </tr>, </td> och </th>. Det är naturligtvis lika viktigt att du inte råkat skriva en avslutningskod för mycket.

Huvudnyckeln till lyckade tabeller är att alltid ha klart för sig skillnaden mellan rader och kolumner. Figur 9.4 visar vad som är vad: raderna löper **horisontellt**, medan kolumnerna löper **vertikalt**. Om du någon gång har svårt att förstå hur en viss tabell kodas, ta då en extra titt på figur 9.4. Sannolikheten är stor att saker och ting då klarnar.



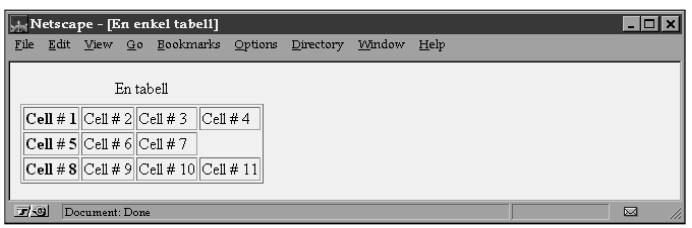
Figur 9.4
För att förstå hur tabeller kodas är det mycket viktigt att skilja mellan tabellens rader och kolumner.

En enkel tabell

Låt oss nu titta på hur en tabell kodas. Knappa in koden i exempel 9.5 a och öppna dokumentet i din webbläsare. Resultatet bör likna figur 9.5 b.

Exempel 9.5 a
En enkel tabell.
Det är bara utseendet som skiljer mellan rubrikceller (<th>) och vanliga celler (<td>). Funktionen är densamma.
Prova gärna att byta ut alla <th> mot <td> och tvärtom.
Resultatet av kodningen syns i figur 9.5 b.

```
1.
2.  <table border>
3.  <caption>En tabell</caption>
4.    <tr>
5.          <th>Cell # 1</th>
6.          <td>Cell # 2</td>
7.          <td>Cell # 3</td>
8.          <td>Cell # 4</td>
9.    </tr>
10.   <tr>
11.         <th>Cell # 5</th>
12.         <td>Cell # 6</td>
13.         <td>Cell # 7</td>
14.         <td></td>
15.   </tr>
16.   <tr>
17.         <th>Cell # 8</th>
18.         <td>Cell # 9</td>
19.         <td>Cell # 10</td>
20.         <td>Cell # 11</td>
21.   </tr>
    </table>
```



Figur 9.5 b
Resultatet.
Lägg märke till att sista cellen i andra raden bara är ett tomt fält utan ram. Det beror på att cellen helt saknar innehåll. Om man vill ha en tom cell med ram måste cellen fyllas med något osynligt, till exempel en radbrytning (<br />).

Kodningen rad för rad

Här följer en genomgång av kodningen i exempel 9.5 a, rad för rad.

1. Tabeller inleds med <table>. För att tabellen ska få en ram används attributet border.

2. Texten mellan <caption> och </caption> blir tabellens rubrik, som automatiskt centreras ovanför själva tabellen.

3. Med koden <tr> startas tabellens första rad.

4–7. Första raden innehåller fyra celler, varav den första är en rubrikcell (<th>). En rubrikcell fungerar precis likadant som en vanlig cell, bortsett från att textinnehållet sätts med fet stil och centreras. Rubrikcellen avslutas med </th> och de vanliga cellerna med </td>. Cellernas textinnehåll placeras mellan varje cells start- och avslutningskod.

8–9. Här avslutas första raden, och den andra inleds.

10–13. En ny rad celler. Sista cellen är tom och visas som ett tomt fält utan ram. Om du vill ha en tom cell med ram räcker det att fylla cellen med ett osynligt tecken, till exempel koden för radbrytning: <br />.

14–20. Andra raden avslutas, tredje inleds och fylls med fyra celler.

21. Tabellen avslutas med </table>

Extra breda celler med colspan

Om vi vill att en eller flera av tabellens celler ska vara större än andra blir kodningen lite mer komplicerad. Då måste vi nämligen få en eller flera celler att sträcka sig över fler än en rad eller kolumn.

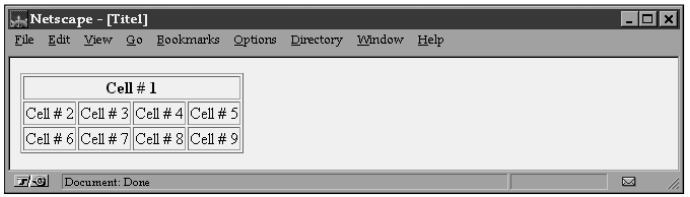
Attributen som åstadkommer detta heter colspan="*antal kolumner*" (kolumnspännvidd) och rowspan="*antal rader*" (radspännvidd).

Kodningen i exempel 9.6 a visar hur colspan används, och resultatet ser du i figur 9.6 b.

Exempel 9.6 a
Olika stora celler.
Första cellen i tabellens första rad sträcker sig över fyra kolumner med hjälp av attributet colspan="4". Resultatet visas i figur 9.6 b på nästa sida.

```
1.  <table border>
2.    <tr>          <! — Första raden —>
3.          <th colspan="4">Cell # 1</th>
4.    </tr>
5.    <tr>          <! — Andra raden —>
6.          <td>Cell # 2</td>
7.          <td>Cell # 3</td>
8.          <td>Cell # 4</td>
9.          <td>Cell # 5</td>
10.   </tr>
11.   <tr>          <! — Tredje raden —>
12.         <td>Cell # 6</td>
13.         <td>Cell # 7</td>
14.         <td>Cell # 8</td>
15.         <td>Cell # 9</td>
16.   </tr>
17. </table>
```

Figur 9.6 b
Resultatet av kodningen i exempel 9.5 a. colspan="4" sträcker ut översta cellen så att den täcker alla fyra kolumnerna i tabellen.



Kodningen rad för rad

Här följer en genomgång av de viktigaste raderna i exempel 9.6 a.

1. Tabellen inleds med <table>.

2–3. Efter att första raden har inletts med <tr> kommer den stora förändringen jämfört med tabellen i exempel 9.5 a och figur 9.5 b: raden innehåller bara en enda cell. Attributet colspan="4" får nämligen cellen att sträcka sig över fyra kolumner. Eftersom kolumner löper vertikalt i tabellen måste cellen bli fyra gånger längre horisontellt för att omfatta fyra kolumner. En <th> eller <td> med attributet colspan blir alltså alltid utsträckt **horisontellt.**

4. Första raden, med den ensamma uttänjda cellen, avslutas med </tr>.

5–17. Resten av tabellen innehåller inga märkvärdigheter. Ytterligare två rader skapas, och dessa fylls med vardera fyra celler. Därefter avslutas tabellen med </table>.

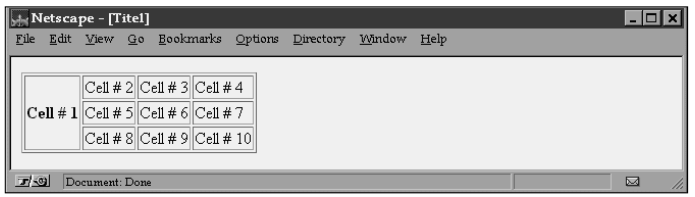
Extra höga celler med rowspan

I ovanstående exempel sträckte vi ut en cell horisontellt. Nu ska vi prova att sträcka en cell vertikalt med attributet rowspan, en procedur som kan vara aningen svårare att förstå sig på än den för colspan. Exempel 9.7 a och b visar hur colspan används.

Exempel 9.7 a
Med hjälp av rowspan="3" dras första cellen ut så att den täcker tre rader. Det innebär att den blir tre gånger så hög som tabellens övriga celler. Resultatet visas i figur 9.7 b på nästa sida.

```
1.  <table border>
2.  <tr>
3.  <th rowspan="3">Cell # 1</th>
4.  <td>Cell # 2</td>
5.  <td>Cell # 3</td>
6.  <td>Cell # 4</td>
7.  </tr>
8.  <tr>
9.  <td>Cell # 5</td>
10. <td>Cell # 6</td>
11. <td>Cell # 7</td>
12. </tr>
13. <tr>
14. <td>Cell # 8</td>
15. <td>Cell # 9</td>
16. <td>Cell # 10</td>
17. </tr>
18. </table>
```


Figur 9.7 b
Resultatet av kodningen i exempel 9.7 a.



Kodningen rad för rad

Här följer en genomgång av de viktigaste raderna i exempel 9.7 a.

1–2. Tabellen startar som vanligt, och första raden inleds.

3. Första cellen är en rubrikcell (<th>) som försetts med attributet rowspan=3. Det betyder att cellen ska dras ut på höjden så att den täcker tre av tabellens rader.

4–7. Ytterligare tre celler fyller ut tabellens första rad, som därefter avslutas med </tr>.

8–12. Tabellens andra rad inleds och fylls med celler. Eftersom tabellens första **kolumn** redan är upptagen av den utdragna rubrikcellen (märkt "Cell # 1") hamnar andra radens första cell (märkt "Cell # 5") i tabellens andra kolumn. Sedan krävs det bara ytterligare två celler för att andra raden ska bli komplett. Därefter avslutas andra raden med </tr>.

13–18. Tabellens tredje rad innehåller också bara tre nya celler, eftersom den utdragna rubrikcellen från första raden ju sträcker sig ända hit. Till sist avslutas sista raden, och därefter tabellen.

Låt dig inte avskräckas

Bli inte avskräckt om du tycker att det är svårt att använda colspan och framför allt rowspan i dina tabeller. Det tycker nästan alla i början. Bästa sättet att lära sig är att experimentera med enkla tabeller och sedan steg för steg öka komplexiteten. Prova till exempel att använda colspan och rowspan mitt inne i en tabell, eller att använda båda attributen i en och samma cell.

Att styra tabellens storlek

Pixlar. Precis som en TV-skärm är datorskärmen uppbyggd av en mängd små punkter – så kallade pixlar. En dators upplösning är det antal pixlar som samtidigt kan visas på skärmen, horisontellt och vertikalt. En upplösning att utgå ifrån när man gör hemsidor är 800 pixlar på bredden och 600 på höjden. Andra upplösningar är 1024 x 768 och i ovanligare fall 1280 x 1024 eller 640 x 480.

Storleken på en standardtabell styrs av hur mycket innehåll det finns i cellerna. Det innebär att den rad och den kolumn som har det mest skrymmande innehållet får bestämma tabellens bredd respektive höjd.

Det finns dock flera olika sätt att själv ta kommandot över tabellernas storlek, och de koderna ska vi titta närmare på nu.

När du bestämmer bredden på en tabell, och kanske därmed hela hemsidan, måste du tänka på att olika datorer har olika upplösning. Ett standardmått för upplösning för en ordinär dator är 800 **pixlar** på bredden och 600 på höjden. Höjden behöver du inte bekymra dig så mycket om. Alla nätsurfare är vana vid att använda rullisterna för att kunna se hela hemsidan. Det kan vara mindre lyckat att låta sidorna överskrida 800 pixlar på bredden. Då kommer dina besökare nämligen att tvingas rulla sidan både vertikalt och horisontellt, och det kommer de garanterat inte att tycka om.

Två attribut styr tabellens storlek: width=”*värde”* eller *värde*% som styr tabellens bredd, och height=”*värde”* eller *värde*% som styr höjden.

Width och height är mycket flexibla attribut. De kan kombineras med såväl <table> som <th> och <td>. Kombinerade med <table> påverkas bredden på tabellen i sin helhet. Kombinerade med <th> (rubrikceller) eller <td> (vanliga celler) påverkar attributen bara storleken på de enskilda cellerna.

Tabellens bredd

I exempel 9.8 a har width använts som attribut till <table>. Det innebär att bredden som anges påverkar tabellen i sin helhet. I det här fallet är tabellen 600 pixlar bred. De tre cellerna får automatiskt en tredjedel var av utrymmet och blir alltså 200 pixlar breda. Resultatet av kodningen visas i figur 9.8 b på nästa sida.

Exempel 9.8 a
En 600 pixlar bred tabell. Resultatet av kodningen visas i figur 9.8 b på nästa sida.

```
<table width="600" border>
<tr>
<th colspan="3">En 600 pixel bred tabell</th>
</tr>
<tr>
<td>Lorem ipsum</td>
<td>Lorem ipsum</td>
<td>Lorem ipsum</td>
</tr>
</table>
```


Figur 9.8 b
Resultatet av kodningen i exempel 9.8 a. Tabellen är 600 pixlar bred, och de tre cellerna får 200 pixlar var.

Du kan även ange tabellens bredd som procent av fönsterbredden, till exempel

```
<table width="50%" border>
```

Tabellen behåller då sin relativa bredd oavsett hur brett användaren väljer att ha sitt fönster.

Cellernas bredd

Ofta är det nödvändigt att ha full kontroll över varje enskild cells storlek. Du kan då ange ett width-värde i varje enskild <td> eller <th>. Se bara till så att summan inte överstiger normal skärmbredd.

Om du anger en bredd i <table> som är mindre än summan av cellernas bredd är det värdet i <table> som gäller. Webbläsaren klämmer då ihop cellerna så mycket som är nödvändigt för att de ska få plats.

Det går också bra att utrycka cellernas bredd i procent av webbläsarens fönster.

Tabellens höjd

Tabellens höjd kan regleras på samma sätt som bredden. Attributet heter height=*värde* eller *värde%* och kan precis som width användas i både <table>, <th> och <td>.

Height kan naturligtvis även kombineras med width, vilket ger dig full kontroll av tabellens storlek. Det går till exempel utmärkt att bestämma tabellens höjd med hjälp av height=”*värde”* i <table> och sedan ange cellernas bredd med hjälp av width=”*värde”* i varje enskild cell.

Att justera tabellens innehåll

I en standardcell blir innehållet vänsterställt och vertikalt centrerat. Detta kan du ändra på med attributen align (efter engelskans *align*, justering) och valign (efter engelskans *vertical align*, vertikal justering). Attributen kan användas tillsammans med <tr>, <th> och <td> och följande värden är tillåtna:

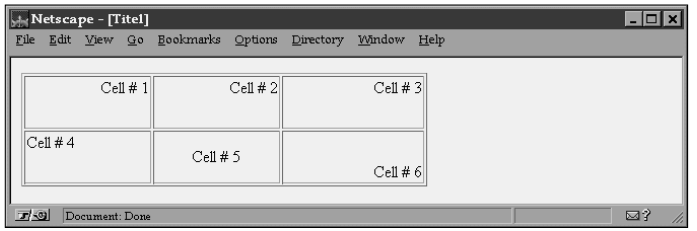
align=	"left" - Innehållet hamnar till vänster
	"right" - Innehållet hamnar till höger
	"center" - Innehållet centreras
valign=	"top" - Innehållet hamnar överst
	"middle" - Innehållet centreras vertikalt
	"bottom" - Innehållet hamnar underst
	"baseline" - Text placeras på en gemensam baslinje

Precis som med height och width blir effekten av attributen olika beroende på vilken kod de används tillsammans med. Tillsammans med <tr> påverkas alla celler i raden (se övre raden i exempel 9.9 a och figur 9.9 b). Om attributen placeras tillsammans med rubrikceller (<th>) eller vanliga celler (<td>) påverkas istället justeringen i varje enskild cell (se nedre raden i figur 9.9 a och b).

Exempel 9.9 a
En tabell med olika typer av justeringar. I första raden har justeringen angivits för hela raden på en gång i <tr>. I andra raden har varje enskild cell fått en egen justering, genom att attributen align och valign kombinerats med <td>. Resultatet av kodningen visas i figur 9.9 b på nästa sida.

```
<table width=400 border>
<tr align="right" valign="top">
<td height=50> Cell # 1</td>
<td height=50>Cell # 2</td>
<td height=50>Cell # 3</td>
</tr>
<tr>
<td height=50 align="left" valign="top">Cell # 4</td>
<td height=50 align="center" valign="middle">Cell # 5 </td>
<td height=50 align="right" valign="bottom">Cell # 6</td>
</tr>
</table>
```


Figur 9.9 b
Resultatet av kodningen i exempel 9.9 a.



Att justera text med baseline

Attributet baseline används när du vill använda olika stor text i tabellens celler. Titta på exempel 9.10 a. Här har vi valt att centrera alla cellernas innehåll vertikalt genom att använda attributet valign="middle" i <tr>. Eftersom texten i de tre cellerna är olika stor hamnar orden inte riktigt på samma rad (figur 9.10 b).

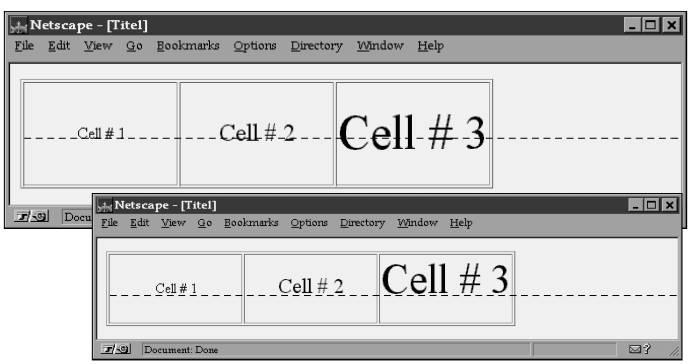
Om vi istället för valign="middle" använder valign="baseline" ser det snyggare ut. All text hamnar då på en gemensam baslinje oavsett storlek (figur 9.10 c).

Exempel 9.10 a
Om olika stor text centreras vertikalt...

```
<table border>
<tr align="center" valign="middle">
<td height=100 width=150><font size=3>Cell # 1 </font></td>
<td height=100 width=150><font size=5>Cell # 2 </font></td>
<td height=100 width=150><font size=7>Cell # 3 </font></td>
</tr>
</table>
```

Figur 9.10 b
... hamnar inte texten på samma rad.

Figur 9.10 c
Men om valign="middle" byts ut mot valign="baseline" ser det bättre ut. Texten hamnar på en gemensam baslinje oavsett storlek.



Ramar och luft

En tabell vars innehåll ser ut att vara inklämt i cellerna är sällan vacker. Med hjälp av attributen cellpadding, cellspacing och border kan du få tabellen att se mer luftig ut. I det här avsnittet ska vi titta närmare på de tre attributen.

Luftiga celler med cellpadding

Cellpadding=”*värde*” är attributet som skapar luft inuti tabellens celler (cellpadding betyder passande nog ”cellstoppning”). Attributet används tillsammans med <table> och värdet är tjockleken på ”stoppningen” mätt i pixlar. En tabell som inleds med

```
<table cellpadding=”6” border>
```

får alltså sex pixlars mellanrum mellan cellernas innehåll och tabellens ram.

Figur 9.11 visar tre tabeller vars cellpadding är satt till två, sex och tio pixlar. Lägg märke till att cellpadding bara kan användas tillsammans med <table>-koden.

Figur 9.11
Attributet cellpadding påverkar hur mycket luft det finns mellan cellernas innehåll och tabellens ram.

Ändra ramtjocklek med border

För att en tabell ska få ram måste attributet border användas tillsammans med <table>. Ramen blir då automatiskt en pixel tjock. Du är dock inte begränsad till en så tunn ram. Genom att komplettera border med ett värde kan du i princip göra ramen hur tjock som helst.

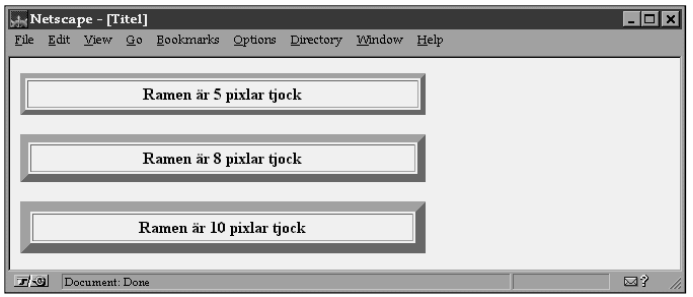
Ofta vill man inte ha någon ram alls på tabellen. Enklaste metoden är att helt enkelt ta bort border från <table>. Ramen blir då osynlig men är fortfarande en pixel tjock. Om du verkligen vill ta bort ramen helt och hållet måste du istället sätta ramtjockleken till noll: border=”0”.

Exempel 9.12 a visar hur du genom att ange border=5 som attribut till <table> skapar en fem pixlar tjock ram. Figur 9.12 b visar resultatet och även hur tabellen ser ut om rambredden ökas till åtta och tio pixlar.

Exempel 9.12 a
Genom att placera attributet border=”*värde*” i <table>-koden styr du ramens tjocklek.

```
<table width=”400” cellpadding=”3” border=”5”>
<tr>
<th>Ramen är 5 pixlar tjock</th>
</tr>
</table>
```

Figur 9.12 b
Resultatet av kodningen i exempel 9.12 a. Här ses också tabeller med åtta och tio pixlar tjocka ramar.



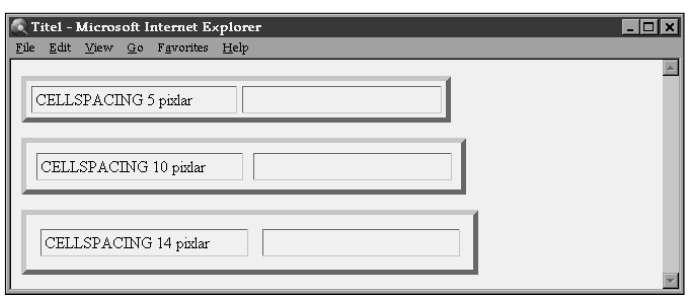
Luft mellan cellerna med cellspacing

Varken cellpadding eller border påverkar avståndet mellan de olika cellerna i tabellen. För detta finns ett särskilt attribut: cellspacing (ungefär ”cellavstånd”). Precis som border och cellpadding är cellspacing ett attribut till <table> och påverkar därmed hela tabellen på en gång. Översta tabellen i figur 9.13 startades så här:

```
<table cellpadding=”3” border=”5” cellspacing=”5”>
```

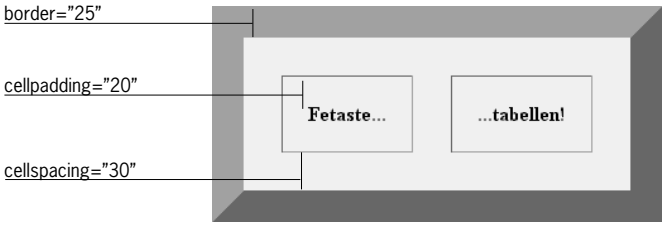
De andra två tabellerna visar hur avståndet mellan cellerna ökar när cellspacing sätts till 10 och 14 pixlar.

Figur 9.13
Attributet cellspacing styr avståndet mellan tabellens celler.



Lathund för ram och luft

Det är lätt att blanda ihop effekterna av border, cellpadding och cellspacing. I lathunden i figur 9.14 ser du vilket attribut som åstadkommer vad.



Figur 9.14
Det tre attributen till <table> som styr tabellens utseende.

Bilder och färg i tabellen

Med hjälp av en tabell kan du få bättre kontroll över bildernas placering på din hemsida. Det är bara att placera bilderna i tabellens celler på samma sätt som du tidigare fyllt cellerna med text.

I exempel 9.15 a har vi gjort en tabell med åtta celler och placerat en bild i varje cell. För att bilderna inte ska hamna direkt intill varandra har tabellen fått en cellspacing på 25 pixlar. När ramen sedan tas bort med attributet border=0 blir resultatet åtta fritt svävande bilder (figur 9.15 b på nästa sida). Koden inklusive alla bilder finns att hämta på bokens hemsida.

Exempel 9.15 a
Åtta bilder sprids ut jämnt på två rader med hjälp av en tabell. Avståndet mellan bilderna styrs av attributet cellspacing="25". Resultatet visas i figur 9.15 b på nästa sida.

```
<table cellpadding="0" border="0" cellspacing="25">
<tr>
  <td><img src="bilder/bild.gif" border="1" /></td>
  <td><img src="bilder/bild2.gif" border="1" /></td>
  <td><img src="bilder/bild3.gif" border="1" /></td>
  <td><img src="bilder/bild4.gif" border="1" /></td>
</tr>
<tr>
  <td><img src="bilder/bild5.gif" border="1" /></td>
  <td><img src="bilder/bild6.gif" border="1" /></td>
  <td><img src="bilder/bild7.gif" border="1" /></td>
  <td><img src="bilder/bild8.gif" border="1" /></td>
</tr>
</table>
```


Figur 9.15 b
Resultatet av kodningen i exempel 9.15 a. Den osynliga tabellen håller bilderna på plats. Avståndet mellan dem är exakt 25 pixlar.

Att ändra tabellens bakgrundsfärg

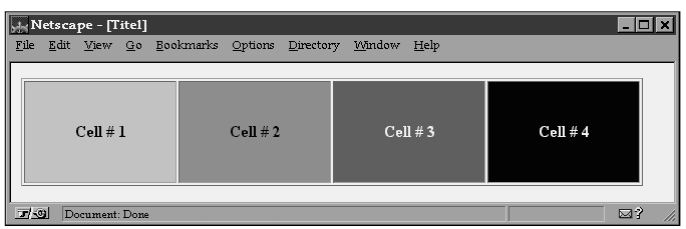
En finess som nu stöds av både Netscape och Internet Explorer är att ge tabellen, en enskild rad i tabellen eller till och med de enskilda cellerna egen bakgrundsfärg. Attributet som åstadkommer detta är detsamma som du använder för att ändra bakgrundsfärg på hela hemsidan: bgcolor="*#färg*", och som alltid i HTML ska färgen specificeras med sexsiffrig hexadecimal kod. Mer om hur färger anges i ett HTML-dokument finns att läsa i kapitel 8.

Bgcolor är ett mångsidigt attribut som kan användas tillsammans med såväl <table> som <tr>, <th> och <td>, med olika resultat. Om bgcolor="#FF0000" placeras i <table> blir hela tabellen rödfärgad. Om en annan färg anges i <tr> får all celler i den aktuella raden denna färg, medan resten av tabellen behåller den röda färg som angavs i <table>. Slutligen kan varje <th> (rubrikcell) och <td> (vanlig cell) förses med sin egen bakgrundsfärg.

I figur 9.16 har tabellens celler (som alla är rubrikceller) fått egna gråtoner. Kodraden som skapar första cellen ser ut så här:

```
<th bgcolor="#CCCCCC" height="100" width="150">Cell # 1</th>
```

Gråtonerna i de övriga tre cellerna är #999999, #666666 och #000000.



Figur 9.16
I den här tabellen har varje cell fått en egen färg. Genom att istället placera bgcolor="*#färg*" i <tr> eller <table> kan en hel rad eller tabellen som helhet förses med en bakgrundsfärg.

Viktigt.
Netscape klarar bara background i <td> och <th>, inte i <table>. Netscape visar inte heller bakgrundsbilder i helt tomma celler. Detta löser du genom att lägga till ** ** (teckenkoden för icke radbrytande mellanslag) eller **
** i tomma celler. Då är cellerna inte längre tomma.

Bakgrundsbilder i tabeller

Länge var det bara Internet Explorer som tillät unika bakgrundsbilder i tabeller, men glädjande nog byggde Netscape in finessen i Netscape Navigator 4.0.

Med hjälp av attributet background=”*sökväg”* anger du vilken bakgrundsbild tabellen, eller en enskild cell, ska ha. Attributet fungerar tillsammans med <table>, <th> (rubrikcell) och <td> (vanlig cell).

Om background=”*sökväg”* används i <table> får samtliga celler i tabellen samma bakgrundsbild. Det går bra att undanta en eller flera celler genom att ge dem en egen background=”*sökväg”*. Hur detta fungerar demonstreras i exempel 9.17 a och figur 9.17 b.

Här har alla tabellens tre celler fått den gemensamma bakgrundsbilden gulsvart.gif genom att background=”*sökväg”* används i <table>. Tabellens andra cell får sedan en unik bakgrundsbild.

Exempel 9.17 a
Med attributet background=”*sökväg”* kan en bakgrundsbild anges för en tabell, en rubrikcell eller en vanlig cell.

```
<table border="0" cellspacing="0" width="95%"
    background="bakgrund/gulsvart.gif">
    <tr>
    <td><br /><br /></td>
    <td background="bilder/filt.gif"><br /><br /></td>
    <td><br /><br /></td></tr>
    </table>
```

Figur 9.17 b
Resultatet av kodningen i exempel 19.17 a.

Tabellens placering på sidan

Precis som bilder kan tabeller göras flytande. Det innebär att de hamnar till höger eller vänster på sidan, och att omkringliggande text flyter runt tabellens vänster- respektive högerkant. Attribut är align=”left” eller *right* och ska placeras tillsammans med <table>. Det är viktigt att komma ihåg att <table align=”right”> påverkar *hela tabellens placering på hemsidan*. När align används som attribut till <tr>, <th> och <td> påverkas däremot justeringen av text och bilder *inne i tabellen.* Attributet align har alltså helt olika innebörd beroende på var det placeras.

Figur 9.18 a
En komplett hemsida som utnyttjar alla tabellkoder. Se sidan i verkligheten på bokens hemsida. Koden till sidan hittar du i exempel 9.18 b.

Alla tabellkoder på en gång

Nu är det dags att kombinera alla länkkoder i en enda tabell. Som exempel ska vi titta på hur den fiktiva hemsidesägaren Alice konstruerat sin sida (figur 9.18 a). Tabellen som styr layouten är ganska avancerad (se exempel 9.18 b). Om du själv vill experimentera finns koden och alla bilder att hämta på bokens hemsida: (www.etc.se/nyckeln/).

Efter koden förklaras de viktigaste momenten.

Exempel 9.18 b
Du behöver naturligtvis inte skriva in hela koden; den och bilderna finns att hämta på bokens hemsida som du når på adress (http://www.etc.se/nyckeln/).

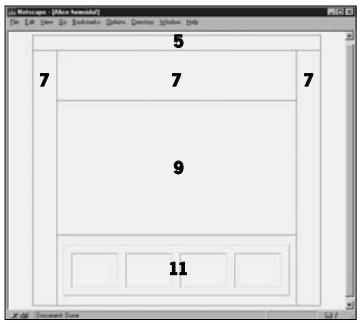
```
1.  <html>
    <head>
    <title>Alice hemsida!</title>
    </head>
    <body bgcolor="#FFFF00">
2.  <center>
3.  <table border="0" bgcolor="#FFFFFF"
    cellpadding="4" cellspacing="0">
4.  <tr>
5.  <td height="5" bgcolor="#FFFF00" colspan="3"> </td>
    </tr>
6.  <tr>
```

```
7.  <td width="38" rowspan="3" valign="top">
    <img src="bilder/vanster.gif" width="38" height="520" /></td>
    <td width="450" valign="middle" align="center"><font
    size="7"><b><tt>Välkommen till Alice
    hemsida!</tt></b></font></td>
    <td width="38" rowspan="3" valign="top"><img
    src="bilder/hoger.gif" width="38" height="520" /></td>
    </tr>
8.  <tr>
9.  <td width="450" valign="top"><img src="bilder/alice.gif"
    align="left" hspace="10" />Lorem ipsum dolor sit amet, consecte-
    tuer adipiscing elit, sed diam nonummy nibh euismod tincidunt
    (...)</td>
    </tr>
10. <tr>
11. <td align="center"><a href="foto.html"><img
    src="bilder/bild5.gif" hspace="10"></a><a
    href="dans.html"><img src="bilder/bild.gif" hspace="10" /></a>
    <a href="resor.html"><img src="bilder/bild10.gif" hspace="10"
    /></a><a href="jobbet.html"><img src="bilder/bild2.gif" hspa-
    ce="10" /></a>
    <br />
    [<a href="foto.html"><font size="2">Foto</font></a>
     l <a href="dans.html"><font size="2">
    Dans</font></a>  l <a href="resor.html"><font
    size="2">Resor </font></a> l 
    <a href="jobbet.html"><font size="2">Jobbet</font></a>]
    </td>
    </tr>
12. </table>
    </center>
    </body>
    </html>
```

Viktiga moment i koden

1. Hela hemsidan får en gul (#FFFF00) bakgrundsfärg.
2. Tabellen centreras med koden .
3. Här startar tabellen. Bakgrundsfärgen sätts till vitt (#FFFFFF) för att tabellen ska skilja sig från resten av hemsidan, som ju är gul.
4. Första raden inleds.

Figur 9.19
Tabellen i figur 9.19, men utan innehåll och med ram. Siffrorna hänvisar till det moment i kodningen där respektive cell skapas.



5. Tabellens första rad innehåller bara en enda cell. Den är till för att skapa luft överst på sidan. Utan denna cell hade tabellens övre kant hamnat allra överst i webbläsarens fönster, vilket givit ett något inklämt intryck. Det enda som finns i själva cellen är det osynliga tecknet för icke radbrytande mellanslag ( ). För att inte den tomma cellen ska synas får den samma färg som hemsidan i övrigt: gult. Attributet colspan=3 gör så att tabellen spänner över tabellens samtliga tre kolumner.
6. Andra raden inleds.
7. Andra raden fylls med tre celler. Vänstra och högra cellen innehåller bilder som föreställer ett datapappers vänster- respektive högerkant (”vanster.gif” och ”hoger.gif”). Bilderna är bara 38 pixlar breda, men hela 520 pixlar höga. Med hjälp av rowspan=”3” tänjs cellerna därför ut över tre rader. Cellen i mitten är 450 pixlar bred och innehåller rubriken ”Välkommen till Alice hemsida!”.
8. Tredje raden inleds.
9. Eftersom kolumnerna till vänster och höger redan är upptagna av de smala, höga cellerna från förra raden, behövs det bara en ny cell för att fylla tredje raden. Cellen innehåller hemsidans huvudtext (något kortad i kodexemplet) och en bild på Alice (alice.gif).
10. Fjärde raden inleds.
11. Då de långsmala cellerna från andra raden fortfarande tar upp tabellens ytterkolumner behöver inte heller den här raden innehålla mer än en ny cell för att bli fylld. Cellen innehåller i sin tur fyra klickbara bilder som ska representera Alice intressen. För att skapa mellanrum mellan bilderna används koden hspace=10, vilken ger varje bild en horisontell marginal på tio pixlar. Alla bilderna är klickbara, och under dem finns samma länkar i textform så att även den som stängt av bildvisningen enkelt ska kunna använda länkarna.
12. Tabellen avslutas.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Förkortningarna till höger anger med vilka webbläsare koderna och attributen fungerar.
ns = Netscape Navigator, ie = Internet Explorer. Siffrorna anger version.

KODER SOM SKAPAR TABELLER

HTML-kod — **Webbläsare**

<table>…</table> — alla

Attribut till <table>

border="*värde*" — alla
Skapar en ram till tabellen. Om inget värde anges blir ramen automatiskt en pixel bred. Om border helt tas bort blir ramen osynlig men tar fortfarande upp en pixels utrymme. border="0" tar bort ramen helt.

cellspacing="*värde*" — alla
Bestämmer avståndet mellan tabellens celler. Om cell spacing helt tas bort blir avståndet automatiskt två pixlar stort. cellspacing="0" innebär att cellerna hamnar omedelbart bredvid varandra.

cellpadding="*värde*" — alla
Bestämmer avståndet mellan ram och innehåll i cellerna. Om cellpadding helt tas bort blir avståndet automatiskt två pixlar stort. cellpadding="0" innebär att cellerna helt fylls ut av innehållet.

width="*värde*" eller "*värde%*" — alla
Bestämmer tabellens bredd, antingen som antal pixlar eller i procent av webbläsarens fönsterbredd.

height="*värde*" eller "*värde%*" — alla
Bestämmer tabellens höjd, antingen som antal pixlar eller i procent av webbläsarens fönsterhöjd.

align="left" eller *right* — alla
Tabellen placeras antingen till vänster (*left*) eller till höger (*right*). Omgivande text flyter sedan runt tabellen.

bgcolor="#*färgkod*" — alla
Anger att tabellen ska ha en annan bakgrundsfärg än övriga hemsidan. Bakgrundsfärgen anges som en sexsiffrig hexadecimal kod.

background="*sökväg till bild*" — ie3-4
Ger alla tabellens celler en och samma bakgrundsbild.

KODER SOM SKAPAR TABELLER (*forts*)

HTML-kod — **Webbläsare**

<tr>…</tr> — alla
Skapar en rad i tabellen.

Attribut till <tr>

align="left", *center* eller *right* — alla
Bestämmer hur texten i radens cell(er) ska vara justerad: till vänster (*left*), centrerad (*center*) eller till höger (*right*). Om align helt tas bort blir radens textinnehåll automatiskt vänster justerat.

valign="*top*", *"middle"*, *"bottom"* eller "*baseline"* — alla
Bestämmer hur texten i radens cell(er) ska vara justerad vertikalt: överst (*top*), i mitten (*middle*), nederst (*bottom*) eller längs samma baslinje (*baseline*).

bgcolor="#*färgkod*" — ie2-4·ns3-4
Anger att radens celler ska ha en annan bakgrundsfärg än övriga hemsidan eller övriga tabellen. Bakgrundsfärgen anges som en sexsiffrig hexadecimal kod.

<th>…</th> och <td>…</td> — alla
Skapar en rubrikcell (<th>) eller en vanlig cell (<td>).

Attribut till <th> och <td>

align="left", *center* eller *right* — alla
Bestämmer hur cellens innehåll ska vara justerat: till vänster (*left*), centrerat (*center*) eller till höger (*right*). Om align helt tas bort blir cellens innehåll automatiskt centrerat.

valign="*top*", *"middle"*, *"bottom"* eller *"baseline"* — alla
Bestämmer hur texten i cellen ska vara justerad vertikalt: överst (*top*), i mitten (*middle*), nederst (*bottom*) eller längs samma baslinje (*baseline*).

height="*värde*" eller "*värde*%" — alla
Bestämmer cellens höjd i pixlar, eller i procent av webbläsarens höjd.

width="*värde*" eller "*värde*%" — alla
Bestämmer cellens bredd, antingen i antal pixlar eller i procent av webbläsarens fönsterbredd.

colspan="*värde*" — alla
Bestämmer hur många kolumner cellen ska sträcka sig över.

rowspan="*värde*" — alla
Bestämmer hur många rader rubrikcellen ska sträcka sig över.

KODER SOM SKAPAR TABELLER (*forts*)

HTML-kod	Webbläsare
bgcolor="*#färgkod*"	ie2-4·ns3-4
Anger att rubrikcellen ska ha en annan bakgrundsfärg än övriga hemsidan, övriga tabellen eller de andra cellerna i samma rad. Bakgrundsfärgen anges som en sexsiffrig hexadecimalkod.	
background="*sökväg till bild*"	ie3-4·ns4
Ger den enskilda cellen en egen bakgrundsbild.	
<caption>…</caption>	alla
Skapar en rubrik till tabellen.	
Attribut till <caption>	
align=*"top"* eller *"bottom"*	alla
Bestämmer om rubriken ska ligga ovanför (*top*) eller under (*bottom*) tabellen. Om align helt tas bort hamnar rubriken automatiskt överst.	

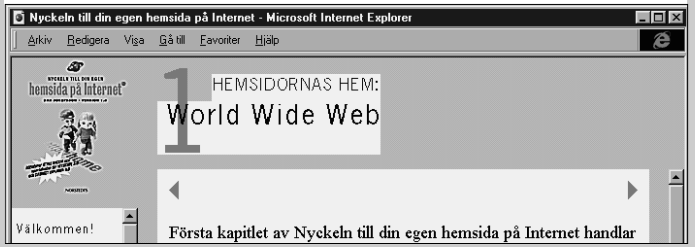
10 VISA FLERA SIDOR PÅ EN GÅNG: Rutsystem

Med rutsystem delar du upp webbläsarens fönster i flera delar som alla kan visa olika hemsidor. Funktionen är mycket kraftfull och kan göra även stora system av hemsidor överskådliga och lättnavigerade. I det här kapitlet lär du dig allt som finns att veta om hur det går till att koda rutsystem.

När rutor (eller *frames* som de kallas på engelska) lanserades med version 2.0 av Netscape Navigator blev det för första gången möjligt att visa mer än en hemsida åt gången i webbläsarens fönster. Trots att ingen annan webbläsare till en början stödde den nya funktionen blev rutanvändandet snabbt allmänt spritt på World Wide Web, och idag ingår rutor i den HTML-specifikation som *World Wide Web Consortium* (*W3C*) sammanställt under namnet HTML 4.0 (du kan läsa mer om W3C och HTML 4.0 i kapitel 2).

Det finns flera fördelar med att använda rutor:

- Rutor gör ditt system av hemsidor mer lättnavigerat och överskådligt.
- Rutor gör det möjligt att låta vissa element, till exempel menyer, ligga kvar när resten av sidan uppdateras. Risken att besökaren missar något blir då mindre.



Figur 10.1
Den här bokens hemsida utnyttjar rutsystem. Den klickbara förteckningen till vänster utgör ett eget HTML-dokument. När du klickar på ett kapitel dyker en ny sida upp i rutan till höger. Lägg märke till att rutornas ramar är osynliga.

Det finns också en del nackdelar med att använda rutor:

• Eftersom varje ruta innehåller en egen hemsida måste webbläsaren kontakta webbservern flera gånger. Detta tar längre tid än om all information legat på en enda hemsida.

• Det är mer krävande för datorn att visa ett rutsystem än en vanlig hemsida. Besökare med långsamma datorer kan därför få problem om ditt system innehåller många rutor.

• Det är svårare att göra hemsidor med rutor än att göra vanliga hemsidor. Framför allt kräver rutor noggrannare planering innan HTML-kodandet börjar.

Figur 10.2
En hemsida som använder rutor. Se sidan i färg på bokens hemsida: www.etc.se/nyckeln/

Ett exempel på rutsystem

För att få grepp om hur rutor fungerar ska vi titta på ett rutsystem som är till för att underlätta läsningen av en elektronisk text, science fiction-klassikern ”A Princess of Mars” av Edgar Rice Burroughs (figur 10.2).

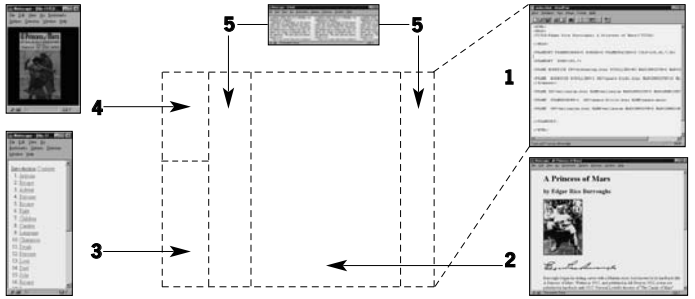
Hemsidan består av inte mindre än sex individuella HTML-dokument. I figur 10.3 (överst på nästa sida) har sidan plockats sönder i sina beståndsdelar. Vi ska gå igenom dessa en och en. Siffrorna hänvisar till siffrorna i figuren.

1. En hemsida som innehåller rutor måste alltid startas från ett speciellt HTML-dokument som har till uppgift att definiera själva rutsystemet, men som i övrigt saknar innehåll. I figuren är rutsystemet markerat med streckade linjer.

2. När rutsystemet är definierat krävs det att varje enskild ruta fylls med HTML-dokument. Den stora rutan i mitten fylls här med en introduktionstext. När användaren klickar i menyn kommer bokens kapitel att visas i detta fönster.

3. Rutan längst ned till vänster innehåller en meny över bokens kapitel. När användaren klickar dyker texten upp i den stora textrutan till höger.

4. Detta dokument innehåller en bild av bokens omslag.



Figur 10.3
Rutsystemet nedplockat i sina beståndsdelar. Lägg märke till att dokumentet längst upp till höger inte har någon annan funktion än att skapa själva rutsystemet (markerat med streckade linjer). Hemsidans innehåll finns i separata HTML-dokument som fyller de enskilda rutorna.

5. De två smala spalterna till höger och vänster om det stora textfönstret innehåller båda ett HTML-dokument som bara innehåller en bakgrundsbild. Därmed är alla rutor fyllda med HTML-dokument, och hemsidan är komplett.

Skapa markeringsdokument

I det här kapitlet kommer du att lära dig allt om hur det går till att använda rutor på en hemsida. Men först måste du skapa de HTML-dokument som vi i övningarna kommer att använda till att fylla rutorna med. Börja därför med att skapa nio olika dokument enligt koden i exempel 10.4. Se till att varje dokument innehåller dokumentets nummer, enligt följande: ”Detta är dokument # 1”, ”Detta är dokumnet # 2” osv till dokument # 9. Döp därefter dokumenten till dok1.html, dok2.html osv till dok9.html och spara dem i den katalog där du förvarar dina övriga HTML-dokument.

Exempel 10.4
Gör nio markeringsdokument enligt den här koden. Spara dokumenten under namnen dok1.html–dok9.html i samma katalog som dina övriga HTML-dokument.

```
<html>
<head><title>Titel</title></head>
<body>
Detta är dokument #1
</body>
</html>
```

Kolumner

Har du sparat dina nio minimala HTML-dokument? Bra, du kommer snart att få användning för dem.

Nu ska vi skapa ett HTML-dokument som definierar ett rutsystem (liknande det som syns överst till höger i figur 10.3).

De HTML-koder som används för ändamålet är:

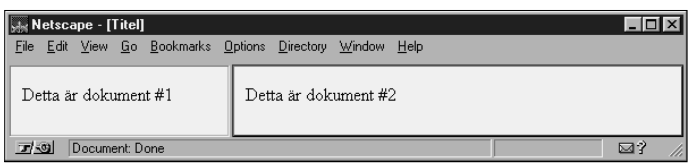
<frameset>...</frameset> som inleder och avslutar rutsystemet och

<frame /> som inleder varje enskild ruta i systemet.

Vi börjar med ett enkelt system som bara består av två kolumner med en ruta i varje. Skapa ett helt nytt dokument och skriv in koden i exempel 10.5 a precis som den står. Om du använder ett malldokument är det mycket viktigt att du raderar koderna <body> och </body>. Dessa får nämligen inte vara med i ett dokument som innehåller <frameset> (mer om detta nedan). Spara dokumentet under valfritt namn, men se till att det hamnar i samma katalog som de nio markeringsdokumenten. När du öppnar dokumentet i en webbläsare kommer det att se ut ungefär som i figur 10.5 b. Nedan går vi igenom koden rad för rad.

Exempel 10.5 a
Notera att <frameset> kommer direkt efter <head>-delen och att <body> helt utgår. Koderna som inleds med "<!" är kommentarer som inte påverkar sidans utseende.

```
    <html>
1   <head><title>Titel</title></head>
2   <frameset cols="200,*">
3           <! En ruta som hamnar i första kolumnen>
            <frame src="dok1.html" />
4
            <! En ruta som hamnar i andra kolumnen>
            <frame src="dok2.html" />
5   </frameset>
6   </html>
7
```



Figur 10.5 b
Sidan består av två kolumner, som innehåller en ruta var. Rutorna är i sin tur fyllda med markeringsdokumenten dok1.html och dok2.html.

Koden rad för rad

1. Precis som ett vanligt HTML-dokument börjar koden med <html>.
2. Därefter kommer HEAD-avsnittet där titeln anges.
3. Här kommer den stora nyheten: <body>-koden har helt utgått.

Dokument som definierar rutsystem är enda undantaget från regeln att alla HTML-dokument ska innehålla koderna <body> och </body>. På rad tre kommer istället koden <frameset>. Den talar om för webbläsaren att ett rutsystem nu inleds. Som du ser har <frameset> också ett attribut: cols="200,*" (efter engelskans *columns*, kolumner). Det talar om att sidan ska ha två kolumner varav den första ska vara 200 pixlar bred och den andra få resten av utrymmet i webbläsarens fönster, vilket markeras med en asterisk (*). Oroa dig inte om du tycker att detta verkar krångligt,

Viktigt.
Ett dokument som innehåller <frameset> får **aldrig** innehålla <body> </body>. Då kommer webbläsaren nämligen att ignorera alla koder som har med rutor att göra, och din sida kommer inte att fungera.

Tips!
Precis som med tabeller är det när man gör rutsystem mycket viktigt att hålla reda på skillnaden mellan *kolumner*, som löper **vertikalt,** och *rader*, som löper **horisontellt**. Titta gärna tillbaka på figur 9.4 i kapitlet om tabeller om du är osäker.

vi kommer snart att grundligt gå igenom hur man bestämmer kolumnernas bredd.

4. Nu är alltså två kolumner definierade, men fortfarande saknas de egentliga rutorna. På fjärde raden skapas den första med koden <frame />. Attributet src="dok1.html" (efter engelskans *source*, källa) talar om att det är HTML-dokumentet "dok1.html" som ska synas i just denna ruta. Lägg märke till att <frame /> avslutas i slutet av själva taggen.
5. Här skapas ytterligare en ruta, som hamnar i andra kolumnen, eftersom den första ju redan är fylld. I denna ruta vill vi visa "dok2.html", vilket vi talar om med attributet src="dok2.html".
6. När båda kolumnerna nu är fyllda med varsin ruta avslutas rutdefinitionen med </frameset>.
7. Sist kan dokumentet som vanligt avslutas med </html>.

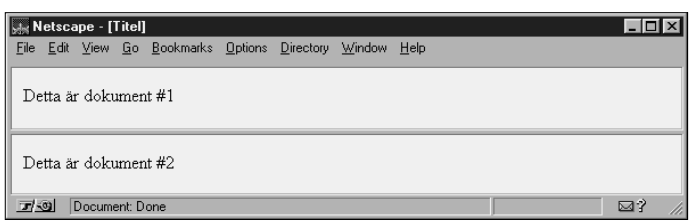
Rader

Rutsystemet ovan består av två kolumner. Om du istället vill ha rader byter du ut attributet cols mot rows (*rader* på svenska). Exempel 10.6 a visar hur detta fungerar.

<frameset rows="60,*"> betyder att första raden ska vara 60 pixlar hög medan den andra får resten av utrymmet i webbläsarens fönster. Figur 10.6 b visar hur det hela ser ut i webbläsaren. Jämför gärna med exempel 10.5 a och figur 10.5 b.

Exempel 10.6 a
Ett rutsystem som består av två rader, med en ruta i varje rad.

```
<html>
<head><title>Titel</title></head>
<frameset rows="60,*">
        <! En ruta som fyller första raden>
        <frame src="dok1.html" />
        <! En ruta som fyller andra raden>
        <frame src="dok2.html" />
</frameset>
</html>
```



Figur 10.6 b
Resultatet av koden i exempel 10.6 a.

Storlek på rader och kolumner

Varje gång du skapar ett nytt rutsystem med koden <frameset> måste du också tala om hur breda respektive höga kolumnerna respektive raderna ska vara. Måtten separeras med kommatecken, och listans längd anger hur många rader eller kolumner det ska vara.

<frameset cols=”100,50,100”> skapar alltså tre kolumner, eftersom det ju innehåller tre värden; 100, 50 och 100. På samma sätt skapar <frameset rows=”100,50,100,25”> fyra rader, eftersom fyra värden angivits.

Det finns tre olika sätt att ange en rads höjd respektive en kolumns bredd. Det enklaste är att ange ett värde mätt i pixlar. <frameset cols=”100,50”> skapar två kolumner som är 100 respektive 50 pixlar breda. Det är enkelt, men kan också vara farligt.

Tänk dig att du skapar ett rutsystem som ser ut så här:

<frameset cols=”150,500,150”>. Sidan blir sammanlagt 800 pixlar bred, vilket är utmärkt för alla som har en skärmupplösning på 800 gånger 600 pixlar. För den majoritet som bara har 640 x 480 pixlar på sina skärmar blir resultatet istället att sidan inte får plats. Deras webbläsare kommer därför att krympa alla de rutor som ingår i systemet för att bredden ska passa skärmupplösningen. Resultatet kan bli förödande, särskilt om en eller flera rutor innehåller HTML-dokument med bilder i. Dessa kan helt eller delvis skymmas när rutorna blir mindre.

Den magiska asterisken

Lyckligtvis finns det ett effektivt sätt att komma ifrån storleksproblemet: genom att byta ut en eller flera storleksangivelser mot en asterisk (*). Asterisken betyder att webbläsaren själv får bestämma storlek på just den kolumn eller rad där asterisken finns, men att alla andra rutor ska behålla sina angivna mått. Genom att använda en eller flera asterisker får du alltså rutsystem som är okänsliga för hur stort webbläsarens fönster är.

Hur fungerar det då i praktiken? <frameset cols=”180,*,180”> talar om för webbläsaren att reservera 180 pixlar för den första och sista kolumnen, och sedan låta kolumnen i mitten få allt det utrymme som blir över. På så sätt kommer mittkolumnen att töjas ut om användaren har en stor skärm, och pressas ihop om skärmen är liten. De yttre kolumnerna kommer dock alltid att behålla 180 pixlars bredd, vilket gör det möjligt att placera till exempel bilder där.

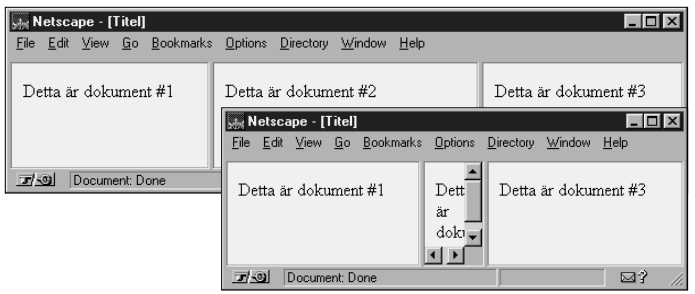
Exempel 10.7 a visar en kod där asterisker används. Figur 10.7 b och c visar hur resultatet blir i en webbläsare med två olika fönsterstorlekar.

Exempel 10.7 a
Här skapas ett rutsystem med tre kolumner. Kolumnen i mitten görs töjbar genom att storleken ersätts med en asterisk.

```
<html>
<head><title>Titel</title></head>
<frameset cols=”180,*,180”>
        <frame src=”dok1.html” />
        <frame src=”dok2.html” />
        <frame src=”dok3.html” />
</frameset>
</html>
```

Figur 10.7 b
De yttre kolumnerna blir 180 pixlar breda. Kolumnen i mitten får utrymmet som blir över.

Figur 10.7 c
När fönstret blir mindre är det bara kolumnen vars bredd angivits med en asterisk som kläms ihop. De yttre kolumnerna förblir 180 pixlar breda.



Det går bra att markera mer än en rad eller kolumn, till exempel:

```
<frameset cols=”180,*,*,180”>
```

Resultatet blir fyra kolumner. De yttre får 180 pixlar var, medan de två i mitten delar på utrymmet som blir över.

En annan variant är att ange hur stor del av det återstående utrymmet varje asteriskförsedd kolumn/rad ska ha. Det åstadkoms genom att man sätter en siffra framför asterisken, till exempel:

```
<frameset cols=”180,2*,*,180”>
```

Kolumnen vars storlek markerats med ”2*” får här dubbelt så mycket av det återstående utrymmet som den andra asteriskförsedda kolumnen.

Det finns, slutligen, möjlighet att ange en rads eller kolumns storlek som procent av webbläsarens fönsterbredd, till exempel:

```
<frameset cols=”25%,50%,25%”>
```

Experimentera gärna med att blanda olika typer av storleksangivelser. Du kommer då att märka att det faktiskt finns flera olika sätt att säga samma sak. <frameset cols=”25%,*,25%”> ger till exempel samma resultat som <frameset cols=”*,2*,*”>.

Att kombinera rader och kolumner

Hittills har vi valt att låta våra rutsystem innehålla antingen rader eller kolumner. Men för att skapa ett rutsystem liknande det i figur 10.2 krävs att vi kombinerar rader och kolumner. När vi i förra kapitlet gjorde tabeller kunde detta åstadkommas med hjälp av koderna rowspan och colspan. När man gör rutsystem är det tyvärr lite mer komplicerat: man måste skapa flera rutsystem i varandra.

Om du tycker att det som nu följer är svårt att förstå är du ursäktad; att kombinera flera rutsystem tillhör det mest komplicerade man kan ta sig för med HTML, och de flesta tycker att det är besvärligt i början. Men om du bara är noggrann och alltid har klart för dig skillnaden mellan en rad och en kolumn kommer du snart att förstå principen.

En webbläsare är synnerligen logisk. När den med koden <frameset cols=”200,*”> får till uppgift att skapa två kolumner, kommer den automatiskt att leta efter två rutor att fylla kolumnerna med. Det har vi bland annat sett exempel på i exempel 10.5 a och figur 10.5 b.

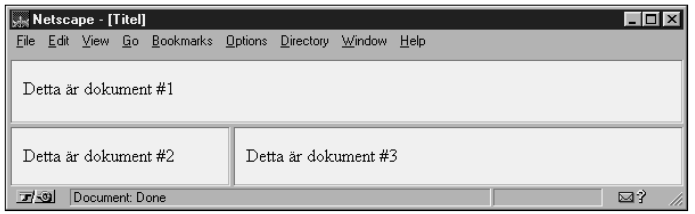
Webbläsaren behöver dock inte nödvändigtvis fylla varje rad eller kolumn med en ruta. Det går lika bra med ett helt nytt rutsystem, vilket gör det möjligt att baka in kolumner inuti rader, eller tvärtom. Jo, det är ganska krångligt, och vi ska därför titta på ett par konkreta exempel.

Öppna ett nytt dokument och skriv in koden i exempel 10.8 a. Spara dokumentet och öppna det med din webbläsare. Det bör se ut ungefär som i figur 10.8 b. Nedan går vi igenom koden rad för rad.

Exempel 10.8 a
Genom att stoppa in ett helt nytt rutsystem i andra raden kan rader och kolumner blandas. På nästa sida går vi igenom koden rad för rad.

```
1  <html>
2  <head><title>Titel</title></head>
3  <frameset rows=”60,*”>
     <! En ruta som hamnar på första raden>
4    <frame src=”dok1.html” />
               <! Ett nytt rutsystem som hamnar på andra raden>
5              <frameset cols=”200,*”>
                         <! En ruta som hamnar i första kolumnen>
6                        <frame src=”dok2.html” />
                         <! En ruta som hamnar i andra kolumnen>
7                        <frame src=”dok3.html” />
8              </frameset>
9  </frameset>
10 </html>
```

Figur 10.8 b
Resultatet av koden i exempel 10.8 a (föregående sida).



Koden rad för rad

1–3. Dokumentet inleds som vanligt. På rad 3 skapas två **rader**. Den ena är 60 pixlar hög och den andra får resten av utrymmet i webbläsarens fönster.

4. Här skapas första rutan. Den hamnar automatiskt i första raden, och fylls med HTML-dokumentet ”dok1.html”.

5. Nu blir det mer avancerat. Eftersom vi vill ha två kolumner i rutsystemets andra rad måste vi fylla hela raden med ett nytt rutsystem som i sin tur innehåller kolumnerna. Vi startar alltså ett helt nytt <frameset>, men den här gången talar vi om att vi vill ha två **kolumner**, varav den första är 200 pixlar bred och den andra får resten av utrymmet. Eftersom webbläsaren ju är på jakt efter något att fylla sin andra rad med, placerar den hela detta nya rutsystem där.

6. Här skapas första rutan i det nya rutsystemet, och den fylls med ”dok2.html”. Webbläsaren placerar denna i första kolumnen.

7. Därefter skapas andra rutan som fylls med ”dok3.html” och placeras i andra kolumnen.

8. Här avslutas det kolumnbaserade rutsystemet.

9. Och slutligen avslutas det radbaserade, inledande rutsystemet.

10. Hemsidan avslutas som vanligt.

Ett exempel till

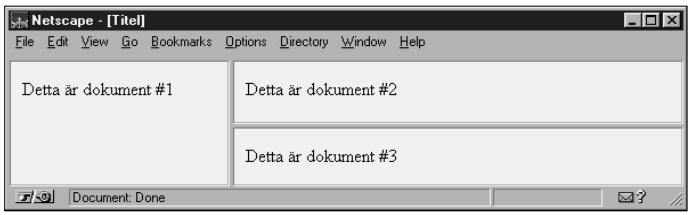
I det här exemplet ska vi inleda med ett kolumnbaserat rutsystem för att sedan spränga in ett baserat på rader. Vi gör alltså precis tvärtom jämfört med exempel 10.8 a och figur 10.8 b. Exempel 10.9 a visar koden och figur 10.9 b resultatet i webbläsarens fönster.

Exempel 10.9 a
Här fylls andra kolumnen med ett nytt, radbaserat rutsystem. Resultatet visas i figur 10.9 b på nästa sida.

```
<html>
<head><title>Titel</title></head>
  <frameset cols=”200,*”>
             <! En ruta som hamnar i första kolumnen>
             <frame src=”dok1.html” />
```

```
<! Ett nytt rutsystem som hamnar i andra kolumnen>
<frameset rows="60,*">
        <! En ruta som hamnar på första raden>
        <frame src="dok2.html" />
        <! En ruta som hamnar på andra raden>
        <frame src="dok3.html" />
</frameset>
</frameset>
</html>
```

Figur 10.9 b
Resultatet av koden i exempel 10.9 a.



En lathund för rutsystem

Att baka in många rutsystem i varandra kan vara förvirrande även för vana HTML-kodare. Du hittar därför en lathund för rutsystem i bilaga 4 längst bak i boken. Om du ska göra ett riktigt komplicerat rutsystem kan du leta upp en grundstruktur i lathunden och sedan modifiera koden för dina behov.

Länkar mellan rutor

En viktig poäng med rutor är att ett länkdokument inte nödvändigtvis måste visas i samma ruta som innehöll den klickbara länken. Att dirigera en länk till en annan ruta kräver två saker:

- att rutan som du vill att länken ska visas i har ett namn.
- att du använder attributet target="*rutans namn*"i själva länken.

För att döpa en ruta använder du attributet name="*rutnamn*" tillsammans med <frame />. Exempel 10.10 visar koden till ett rutsystem med två rutor, där den nedre rutan fått namnet "nedre_rutan". Knappa in koden och spara dokumentet. En av rutorna ska fyllas med ett dokument som ännu inte finns: "meny.html". Det återkommer vi strax till.

Tips!
Så här stegar du fram och tillbaka i rutor:
Internet Explorer
Markera den ruta där bläddringen ska ske genom att klicka i den och klicka först därefter på framåt- eller bakåtknappen.
Netscape
Högerklicka (Windows) eller håll ned musknappen (Mac) i den aktuella rutan. Välj sedan "back" eller "forward".

Exempel 10.10
Rutsystemet innehåller två rader. Den nedersta har fått namnet "nedre_rutan" med hjälp av attributet NAME.

```
<html>
<head><title>Titel</title></head>
        <frameset rows="150,*">
        <frame src="meny.html" />
        <frame name="nedre_rutan" src="dok1.html" />
        </frameset>
</html>
```

Lägg märke till att namnet på rutan "nedre_rutan" är ett enda ord. Om du vill använda namn med mellanslag, till exempel "nedre rutan" måste namnet stå inom citationstecken: name="nedre rutan".

Rutorna i koden ovan är fyllda med två dokument: meny.html överst, och dok1.html underst.

Exempel 10.11 visar koden till "meny.html". Knappa in den i ett nytt dokument och glöm inte att spara under namnet "meny.html".

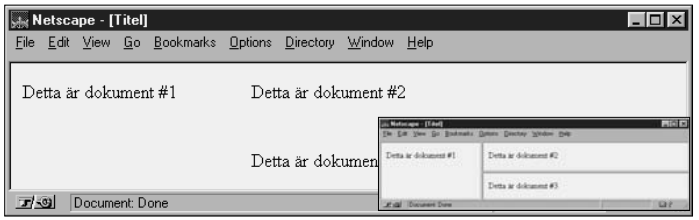
Lägg märke till att alla länkar har "nedre_rutan" som target. Det innebär att länkdokumenten kommer att hamna i just "nedre_rutan" när användaren klickar på en länk.

Exempel 10.11
Den här koden ska sparas under namnet "meny.html".

```
<html>
<head><title>Titel</title></head>
<body>
Vilket dokument vill du se i nedre rutan?<P>
<a href="dok1.html" target="nedre_rutan">Dokument
#1</a><br />
<a href="dok2.html" target="nedre_rutan">Dokument
#2</a><br />
<a href="dok3.html" target="nedre_rutan">Dokument
#3</a><br />
<a href="dok4.html" target="nedre_rutan">Dokument #4</a>
</body>
</html>
```

Öppna nu dokumentet som innehåller koden från exempel 10.10. Om du sparat dokumentet "meny.html" under rätt namn och i samma katalog som de övriga dokumenten kommer länkarna i övre rutan att leda till den nedre.

Experimentera gärna med att skapa länkar mellan rutorna i större rutsystem.



Figur 10.12 a
Genom att göra ramarna mellan rutorna osynliga går det inte att direkt se på hemsidan att den innehåller rutor.

Figur 10.12 b (lilla bilden) visar hur samma rutsystem ser ut med ramarna synliga.

Rutsystem utan ramar

Hittills har vi bara använt två attribut till <frameset>: cols="*kolumnbredd"* och rows="*radhöjd"*. Kvar har vi två attribut som är till för att ta bort rutsystemets ramar: "frameborder=*1* eller *0"* och "framespecing=*värde"*.

Frameborder har bara två möjliga värden: 1, som innebär att ramarna mellan rutorna är synliga (vilket är den automatiska inställningen), och 0, som gör ramarna osynliga. En osynlig ram tar fortfarande upp plats. Om du helt vill ha bort ramen måste du dessutom använda framespacing=0. Nu hamnar rutorna omedelbart intill varandra. Båda attributen påverkar samtliga rutor i rutsystemet, och dessutom alla nya <frameset> som eventuellt fyller rutorna. Du behöver alltså bara ange frameborder och framespacing i systemets första <frameset>.

Med framespacing="*värde"* kan du naturligtvis också göra extra tjocka ramar mellan rutorna.

Rambråket mellan Microsoft och Netscape

Det var Microsoft som uppfann koderna som tar bort ramarna i ett rutsystem. Netscape byggde in funktionen i den slutgiltiga versionen av Netscape Navigator 3.0. Idag fungerar därför attributet frameborder=*0* eller *1* med båda läsarna. Problemet är att Netscape Navigator fortfarande inte förstår attributet framespacing="*värde"*. För att få bort avståndet mellan rutorna i Netscape ska av någon anledning attributet border=0 istället användas.

Den här missen från Netscapes sida rör till det lite grann för dig som vill göra ramlösa rutsystem. Enda sättet att göra ett rutsystem där rutorna ligger direkt intill varandra utan ramar i både Internet Explorer och Netscape är att använda **både** attributet framespacing="0" och border="0". Så här kan ett sådan rutsystem inledas:

```
<frameset frameborder="0" framespacing="0"
border="0" rows="150,*">
```

Viktigt.
För att ramarna på dina rutor ska bli osynliga i både Netscape Navigator och Internet Explorer **måste** både frameborder, framespacing och border sättas till 0.

I denna kod kommer Netscape Navigator att förstå border="0" men ignorera framespacing="0". Internet Explorer gör tvärtom. Det är troligt att framespacing="*värde*" kommer att fungera med framtida versioner av Netscape Navigator. Men fram till dess är det i vanlig ordning vi HTML-kodare som får ta konsekvenserna av att Netscape och Microsoft inte talar med varandra.

Att styra enskilda rutor

Medan attribut till <frameset> påverkar samtliga rutor i ett system, påverkar attribut till <frame /> bara den enskilda rutan. Vi har redan gått igenom två sådana attribut: src="*sökväg*", som talar om vilket HTML-dokument som ska fylla den enskilda rutan, och name="*namn*", som ger rutan ett namn.

Här är resten av attributen:

frameborder="*0*" eller "*1*"
Det här attributet styr huruvuda en enskild ruta ska ha ram eller ej. Även om du gjort alla ramar i systemet osynliga med hjälp av frameborder=*0* i <frameset>, kan du använda frameborder="*1"* i en eller flera enskilda rutor. Ramen kommer då bara att synas kring dessa rutor.

marginwidth="*värde"*
Skapar en vänster- och en högermarginal mellan ramen och det HTML-dokument som fyller ramen. Värdet mäts i antal pixlar. Detta attribut är särskilt praktiskt att använda om HTML-dokumentet som ska visas i ramen innehåller text. Genom att till exempel ange marginwidth="*20"* hamnar texten 20 pixlar från vänster respektive höger kant, vilket gör den lättare att läsa.

marginheight="*värde"*
Samma sak som marginwidth, men kontrollerar istället den övre och undre marginalen. Även detta attribut lämpar sig väl för rutor som ska innehålla HTML-dokument med mycket text. Det blir sällan snyggt om texten börjar allra överst i fönstret. marginheight="*20"* flyttar ned hela HTML-dokumentet 20 pixlar.

scrolling="*yes", "no"* eller "*auto"*
Om ett HTML-dokument är för stort för den ruta det visas i är det viktigt att rutan har rullister, annars kan användaren inte se hela innehållet. Genom att ange scrolling="*yes*" får rutan alltid rullister, oavsett om innehållet får plats eller ej. Bättre är att skriva scrolling="*auto*". Det innebär att rullisterna inte visas förrän de verkligen behövs. scrolling="*no*" innebär att rullisterna aldrig visas,

oavsett om innehållet får plats i rutan eller ej. Denna inställning är bra att använda om rutan till exempel bara ska innehålla en enda bild, som du vet får plats i fönstret.

noresize

Användaren kan normalt ändra storleken på en ruta genom att med musen flytta ramen till ett nytt läge. Om noresize används förhindras detta. Det är lämpligt att använda noresize-attributet i rutor som ska innehålla bilder och annan information som kräver en fast rutstorlek.

Om inte webbläsaren klarar rutor

Det finns fortfarande en del äldre webbläsare som inte klarar att visa rutor. När en sådan läsare stöter på ett dokument som innehåller koderna <frameset> och <frame /> förstår den överhuvudtaget inte vad koderna betyder, och ignorerar därför hela HTML-dokumentet.

Det finns två lösningar på detta problem.

Den vanligaste metoden är att göra två olika hemsidor; en med rutor och en utan, och sedan låta besökarna välja vilken version de vill titta på, beroende på vilken webbläsare de har.

En lite snyggare metod är att i det dokument som definierar rutsystemet också använda koden <noframes>…</noframes>.

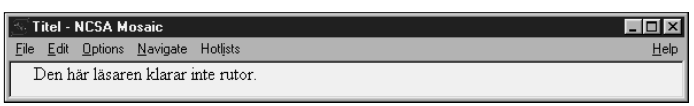
Den läsare som inte klarar rutor använder istället de HTML-koder som placerats mellan <noframes> och </noframes>. Webbläsare som klarar rutor bryr sig däremot inte alls om extrakodningen.

Figur 10.12 a visar en hemsida som innehåller dels ett rutsystem, dels en text placerad mellan <noframes> och </noframes>. Figur 10.12 b visar hur sidan ser ut i Mosaic – ett exempel på webbläsare som inte stödjer rutsystem.

Exempel 10.13 a
Texten eller HTML-koden mellan <noframes> och </noframes> läses bara av webbläsare som inte klarar rutsystem.

```
<frameset rows=60,*>
        <frame src=dok1.html />
        <frame src=dok2.html />
  </frameset>
<noframes>
Den här läsaren klarar inte rutor.
</noframes>
```

Figur 10.13 b
Resultatet av koden i exempel 10.13 a, sett med Mosaic, som inte klarar rutsystem.



Det är fritt fram att lägga in HTML-kod, eller till och med en komplett hemsida, mellan <noframes> och </noframes>. Besökare som har äldre versioner av webbläsare kommer då bara att se den alternativa sidan, och inte ens märka att han eller hon missat något.

Eftersom inte heller äldre versioner av Netscape och Internet Explorer klarar rutsystem, är det klokt att alltid inkludera alternativ information med hjälp av <noframes>, även om det bara är en text som uppmanar besökaren att skaffa en modernare webbläsare.

Koderna i sammanfattning

Här följer en sammanfattning av alla koder i kapitlet. Förkortningarna till höger anger med vilka webbläsare koderna och attributen fungerar.
ns = Netscape Navigator, ie = Internet Explorer. Siffrorna anger version.

KODER SOM SKAPAR RUTSYSTEM OCH RUTOR

HTML-kod	Webbläsare
<frameset>…</frameset>	ie3-4·ns2-4
Skapar ett rutsystem.	
Attribut till <frameset>	
rows="*värde*", *n*% eller *n**	ie3-4·ns2-4
Bestämmer antalet rader, och hur höga dessa ska vara.	
cols="*värde*", *n*% eller *n**	ie3-4·ns2-4
Bestämmer antalet kolumner, och hur breda dessa ska vara.	
frameborder="*0*" eller "*1*"	ie3-4·ns2-4
Bestämmer om systemets samtliga rutor ska ha ram eller ej.	
framespacing="*värde*"	ie3-4·ns2-4
Bestämmer avståndet mellan rutorna (Internet Explorer).	
border="*värde*"	ie3-4·ns2-4
Bestämmer avståndet mellan rutorna (Netscape Navigator).	
<frame />	ie3-4·ns2-4
Skapar en ruta.	
Attribut till <frame />	
src="*adress*"	ie3-4·ns2-4
Bestämmer vilket HTML-dokument som ska visas i rutan.	
name="*rutnamn*"	ie3-4·ns2-4
Ger rutan ett namn.	
marginwidth="*värde*"	ie3-4·ns2-4
Skapar en vänster- och en högermarginal mellan rutans innehåll och ramen.	

HTML-kod	Webbläsare
marginheight="*värde*"	ie3-4·ns2-4
Skapar en övre och en undre marginal.	
frameborder="*0*" eller "*1*"	ie3-4·ns2-4
Bestämmer om rutan ska ha ram eller ej.	
scrolling="*yes*", *"no"* eller *"auto"*	ie3-4·ns2-4
Avgör om rutan ska ha en rullist eller inte.	
noresize	ie3-4·ns2-4
Gör så att ramarna inte är flyttbara.	
<noframes>…</noframes>	ie3-4·ns2-4
Ger dig möjlighet att lägga in HTML-kod som bara läses av webbläsare som inte klarar rutsystem.	

LÄNKAR MELLAN RUTOR

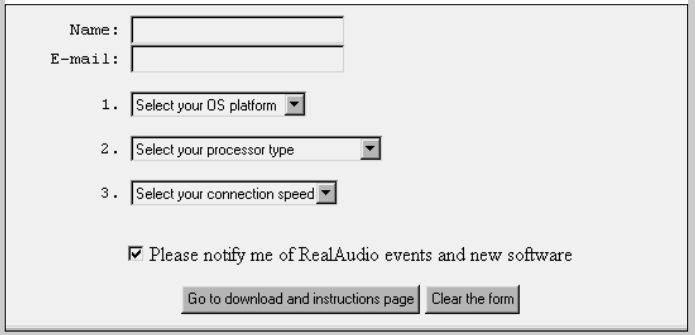
<a>…	ie3-4·ns2-4
Skapar en länk. Måste användas tillsammans med href="*sökväg*".	
Attribut till <a> i samband med rutor	
target="*fönster-* eller *rutnamn*".	ie3-4·ns2-4
Skickar länken till den namngivna rutan.	

11 FÖR INTERAKTIVA HEMSIDOR: Formulär

Om du vill att dina besökare ska svara på enkäter, skicka brev, göra beställningar eller på annat sätt kommunicera med dig är det praktiskt att utrusta hemsidan med formulär. I det här kapitlet får du veta allt om de många formuläralternativ som finns att välja mellan.

Figur 11.1 visar en del av företaget Progressiv Networks hemsida. Här kan besökaren gratis ladda ned plug-in-modulen ”RealAudio”, som gör det möjligt att ta emot direktsänt ljud via Internet. För att få modulen måste man dock fylla i namn, e-postadress och önskad produkt i ett *formulär*. När användaren klickar på den stora knappen nederst (*sändknappen*) skickas uppgifterna till Progressiv Networks webserver, och användaren får hoppa vidare till nedladdningssidan.

Det finns två olika sätt att ta hand om informationen från ett formulär. Antingen skickas den i form av e-post till den som gjort hemsidan. Eller, vilket är mer avancerat, sänds den till ett särskilt så kallat *CGI-program* på webservern för vidare bearbetning. CGI-programmet kan göra allt från att



Figur 11.1
Ett exempel på *formulär*. Användaren fyller i rutorna och väljer alternativ i menyerna. När han eller hon sedan trycker på *sändknappen* nederst till vänster skickas uppgifterna till den som gjort hemsidan.

organisera informationen i en databas till att generera nya hemsidor utifrån de val användaren gjort. I kapitel 13 får du veta mer om CGI-program. I detta kapitel kommer vi att utnyttja det enklaste sättet att ta hand om information från ett formulär: via vanlig e-post.

Formulärets uppbyggnad

Ett formulär består av start- och avslutningskoderna <form> respektive </form>, och ett valfritt antal *formulärobjekt* som placeras mellan start- och avslutningskoderna. Formulärobjekt kan vara utformade på många olika sätt, från kryssrutor till menyer och stora textfält. Detta återkommer vi strax till. Nu ska vi titta närmare på startkoden.

<form>-koden

Startkoden <form> har tre attribut som tillsammans avgör vart och på vilket sätt informationen i formuläret ska skickas när användaren klickar på sändknappen. Här är alla attributen:

action="*adress"*
Om innehållet i formuläret ska skickas till ett CGI-program eller t.ex. en php- eller asp-sida anges adressen till programmet eller sidan här. Om programmet heter "formmail.pl" och ligger i "cgi-bin"-katalogen på webservern (du får veta mer om denna katalog i kapitlet om CGI-script) ska action-attributet se ut så här:

```
action="/cgi-bin/formmail.pl"
```

Om innehållet ska skickas som e-post anges istället användarens e-postadress med hjälp av "mailto:"-funktionen. Om brevet ska skickas till Alice som har e-postadressen alice@internet.se ska action-attributet se ut så här:

```
action="mailto:alice@internet.se"
```

Viktigt.
Kontrollera noga att inga mellanslag smyger sig in i attributen. action="mailto:alice@internet.se" måste till exempel skrivas som ett ord.

method="*get"* eller *"post"*
Här anges på vilket sätt informationen ska skickas. method="get" är ett speciellt sätt att skicka informationen till vissa applikationer. method="post" är dock den vanligaste metoden, och den som alltid används då informationen ska skickas med e-post.

enctype="*text/plain"*
Det här attributet talar om att informationen ska formateras som vanlig text, och ska alltid vara med i formulär som skickas som e-post.

Om du skickar informationen till ett CGI-program behöver du inte ha med enctype-attributet. Då blir enctype-inställningen nämligen automatiskt "application/x-www-form-urlencoded", vilket innebär att specialtecken som radmatningar, svenska tecken och liknande kodas på ett sätt som CGI-program förstår. För att förstå hur <form> fungerar ska vi göra ett enkelt formulär. Om du vill prova att skicka resultatet som e-post till dig själv måste du koppla upp dig mot din Internetleverantör innan du provkör formuläret. Du måste också använda Netscape Navigator 3.0 eller 4.0, eftersom det för närvarande är den enda webläsare som klarar att formatera informationen så att den blir läsbar via e-post.

Knappa in koden i exempel 11.2 a i HTML-dokumentets <body>-del (byt ut Alice adress mot din). Spara dokumentet och öppna det i Netscape. Resultatet bör likna figur 11.2 b. Nedan går vi igenom koden rad för rad.

Exempel 11.2 a
Ett enkelt formulär som består av två formulärobjekt: en textruta och en sändknapp. Du måste naturligtvis byta ut alice@internet.se mot din egen e-postadress. Resultatet ser du i figur 11.2 b nedan.

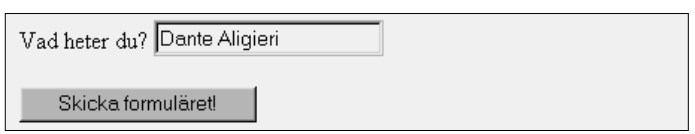
```
1 <form action="mailto:alice@internet.se" method="post"
  enctype="text/plain">
2 Vad heter du? <input type="text" name="Jag heter" /><p>
3 <input type="submit" value="Skicka formuläret!" /></p>
4 </form>
```

Koden rad för rad

1. Formuläret inleds med startkoden <form>. Som action använder vi "mailto:alice@internet.se", vilket innebär att informationen i formuläret kommer att skickas till Alice e-postadress. method="post" talar om att brevet ska skickas som e-post, och med enctype="text/plain" talar vi slutligen om att texten ska formateras så att den blir läslig i e-postform.
2. Den här raden inleds med en text som frågar användaren vad han eller hon heter. Därefter följer första formulärobjektet, och med attributet type="text" talar vi om att vi vill ha en textruta.

   Varje formulärobjekt måste ha ett namn. Eftersom den här textrutan frågar efter användarens namn kallar vi den name="Jag heter".
3. Nu kommer ytterligare en <input />, men av ett helt annat slag. type="submit" (det engelska ordet för "avge" eller "lämna in") skapar en knapp som när användaren klickar på den skickar iväg informationen i formuläret till den adress, och på det sätt, som vi angivit i <form>. value="Skicka formuläret!" talar om vad det ska stå på knappen.
4. Här avslutas formuläret.

Figur 11.2 b
Resultatet av kodningen i exempel 11.2 a. När du fyllt i ett namn i rutan kan du koppla upp dig.



Provskicka ditt formulär

Fyll nu i något i textrutan och klicka på sändknappen. Om du skrivit in din egen e-postadress efter ”mailto:” i koden tar det bara någon sekund innan brevet är framme i din brevlåda. Starta därför ditt e-postprogram och hämta brevet. Det kommer att se ut ungefär som i figur 11.3.

Figur 11.3
Så här ser brevet med uppgifterna från formuläret ut. Till vänster står formulärobjektets namn, ”Jag heter”, och till höger värdet som användaren knappat in, ”Dante Aligieri”.

Lägg märke till att namnen som du tilldelade formulärobjekten med attributet name=”Jag heter” står till vänster i brevet, och att den inskrivna texten (vilket brukar kallas formulärobjektets *värde*) står till höger. När du gör långa formulär kommer resultatet att bli en lång lista av formulärnamn parade med värden.

Formulärobjekten

Exemplet ovan innehåller bara två formulärobjekt, men naturligtvis kan du ta med så många du behöver i ditt formulär. Det finns tre huvudtyper som du kan blanda fritt mellan <form> och </form>:

- <input /> finns i flera modeller.
- <textarea>...</textarea> skapar en stor textruta.
- <select>...</select> skapar en meny.

<input />

<input>-koden måste kompletteras med attributet type=”*checkbox”, “radio”, “text”, “password”, “hidden”, “submit”, “reset”, “button”* eller *“image”*. Vi ska titta på vart och ett av alternativen för sig.

Kryssruta (type=”checkbox”)

Detta attribut skapar små rutor för användaren att kryssa i. Flera altenativ kan kryssas i åt gången. Exempel:

```
Vilka är dina favoritfrukter?<br />
<input type=”checkbox” name=”Favorit: äpple” />Äpplen<br />
<input type=”checkbox” name=”Favorit: banan” />Bananer<br />
<input type=”checkbox” name=”Favorit: päron” />Päron
```


En ikryssad ruta får värdet ”on” när formuläret skickas. Resultatet av ovanstående ikryssningar blir alltså det här i e-brevet:

```
Favorit: äpple=on
Favorit: päron=on
```

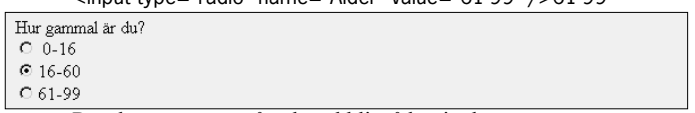
Om du vill att en eller flera kryssrutor ska vara ikryssade redan när formuläret laddas lägger du till attributet checked, till exempel:

```
<input type=”checkbox” name=”Favorit: päron”
checked=”checked” />
```

Radioknappar (type=”radio”)

Radioknappar är bra när användaren bara får välja ett av flera alternativ. Du kan rada upp hur många som helst efter varandra. Exempel:

```
Hur gammal är du?<br />
<input type=”radio” name=”Ålder” value=”0-16” /> 0-16<br />
<input type=”radio” name=”Ålder” value=”16-60” />16-60<br />
<input type=”radio” name=”Ålder” value=”61-99” />61-99
```



Resultatet av ovanstående val blir så här i e-brevet:

```
Ålder=16-60
```

Textrutor (type=”text”)

Du har redan sett en textruta i aktion på andra raden i exempel 11.2 a. Med attributet size=”*antal tecken”* kan du bestämma hur lång rutan ska vara mätt i tecken. Med attributet maxlength=”*antal tecken”* begränsar du det antal tecken som användaren får skriva in i rutan. Genom att ange value=”*valfri text*” kan du slutligen visa en förvald text i rutan, som kommer att skickas som textrutans värde om inte användaren ändrar den.

Dold text-ruta (type=”password”)

Fungerar som en textruta, med den skillnaden att texten som skrivs in markeras med asterisker (*) istället för klartext. Texten skickas dock som vanlig, okrypterad text. Exempel:

```
Lösenord? <input type=”password” name=”Lösenord” />
```

Dold information (type="hidden")

Ibland är det bra att kunna skicka med lite egen information till CGI-programmet, sidan eller e-brevet utan att användaren vet om det. Om du har många olika formulär på dina sidor kan du till exempel lägga till dold information om vilket formulär användaren fyllt i. Exempel:

```
<input type="hidden" name="Formulär nummer"  value="10" />
```

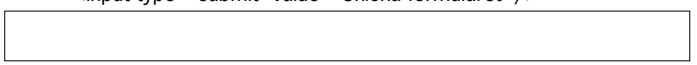
I e-brevet blir resultatet så här:

```
Formulär nummer=10
```

Sändknapp (type="submit")

Alla formulär måste innehålla en knapp med vilken användaren skickar iväg formulärinformationen. Med attributet value=*"text*" bestämmer du vad som ska stå på knappen. Exempel:

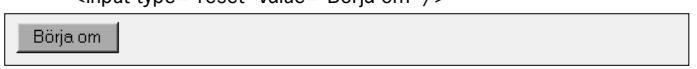
```
<input type="submit" value="Skicka formuläret" />
```



Återställningsknapp (type="reset")

Ett klick på återställningsknappen återställer formuläret till sitt ursprungliga utseende så att användaren kan börja om från början med ifyllandet. Knappen behövs egentligen bara när du gör riktigt stora formulär. Med value="*text"* bestämmer du vad som ska stå på knappen.

```
<input type="reset" value="Börja om" />
```



Bilder (type="image")

Den här <input>-typen låter dig använda en bild som sändknapp. När användaren klickar på bilden skickas alltså hela formuläret iväg, men också koordinaterna för den punkt i bilden användaren klickade på. Du kan använda align="*top", "middle"* eller "*bottom"* för att justera bilden i förhållande till textraden. Exempel:

```
<input type="image" name="bild" src="bilder/minibild.gif" />
```

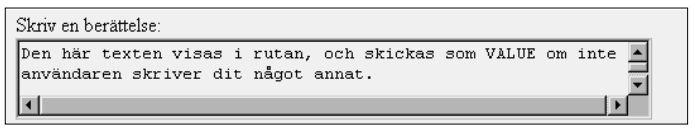
I e-brevet visas koordinaterna för klickningen så här:

```
bild.x=12
bild.y=16
```


Textfält med <textarea>...</textarea>

<input type="text" /> är ett bra formulärobjekt när användaren ska skriva in sitt namn eller några få ord. För längre texter är dock koden <textarea> bättre. Med den skapar du ett textfält med valfri höjd och bredd. Hur många bokstäver som ska få plats i textfältet anges med attributen cols="*värde"* (bredd räknat i antal bokstäver) och rows="*värde"* (höjd i bokstäver). All text som finns mellan <textarea> och </textarea> hamnar inuti textfältet då formuläret laddas, och skickas som värde om inte användaren skriver dit något annat. Exempel:

```
Skriv en berättelse:<br />
<textarea name="Berättelse" rows="2" cols="60">
Den här texten visas i rutan, och skickas som value om inte
användaren skriver dit något annat.
</textarea>
```



Menyer med <select>...</select>

Menyer inleds med <select>, och med attributet size="*värde"* anges hur många alternativ menyn ska innehålla. Om size="1" blir menyn en popup-meny (liknande menyerna i figur 11.1). Attributet multiple gör att flera alternativ kan väljas. Varje alternativ inleds sedan med <option>, som avslutas med </option>. Till sist avslutas menyn med </select>. Exempel:

```
Välj ett eller flera favoritdjur:
<select name="Favoritdjur" multiple size="3">
<option value="Ko">Kossor</option>
<option value="Gris">Grisar</option>
<option value="Lama" selected="selected">Lamadjur</option>
</select>
```

Ovanstående val ger följande resultat i e-brevet:


```
Favoritdjur=Ko
Favoritdjur=Lama
```

Koderna i sammanfattning

Formulär-koder
Samtliga koder fungerar med Netscape Navigator 2.0-4.0 (eller senare) och Internet Explorer 2.0-4.0 (eller senare).

HTML-kod

<form>…</form>
Skapar och avslutar ett formulär.

Attribut till <form>

action="*adress*"
Talar om vart formulärets innehåll ska skickas.
method="*get*" eller "*post*"
Talar om hur innehållet ska skickas.

<input />
Skapar ett formulärobjekt.

Attribut till <input />

type="*checkbox*", *"radio"*, *"text"*, *"password"*, *"hidden"*, *"submit"*, *"reset"* eller "*image*". Anger typ av <input />.
checked=*"checked"*
Förväljer en kryssruta eller en radioknapp.
maxlength="*värde*"
Anger maximalt antal tecken som får skrivas i en textruta.
name="*namn*"
Ger formulärobjektet ett namn.
size="*värde*"
Anger en textrutas längd mätt i antal tecken.
value="*text*" eller *"värde"*
Ger formulärobjektet ett förvalt värde.

<textarea>...</textarea>
Skapar ett textfält.

Attribut till <textarea>

cols="*värde*"
Bestämmer textfältets bredd mätt i antal tecken.
rows="*värde*"
Bestämmer textfältets höjd mätt i antal tecken.

<select>…</select>
Skapar en meny med flera val.

Attribut till <select>

size="*värde*"
Anger menyns längd. size="1" skapar en popup-meny.
multiple=*"multiple"*
Tillåter att fler än ett alternativ kan väljas i menyn.

<option>...</option>
Skapar ett val i selectmenyn.

NÄR SIDAN SKA UT PÅ NÄTET:

12 Uppladdning

För att människor runt om i världen ska kunna besöka din hemsida räcker det naturligtvis inte att den ligger på din egen dator – den måste laddas upp till webservern. I det här kapitlet går vi igenom hela uppladdningsprocessen och ger tips om användbara program för ändamålet.

Nu har stunden kommit till sista momentet i processen att göra en hemsida: uppladdningen. I det här kapitlet lär du dig tre saker:

- Att koppla upp dig mot din Internetleverantörs webserver med ett *FTP-program.*
- Att skapa en särskild katalog för dina hemsidor och kopiera alla dina hemsidor till katalogen.
- Att sätta rätt behörighet på hemsidorna så att de blir tillgängliga för besökare.

Skaffa ett FTP-program

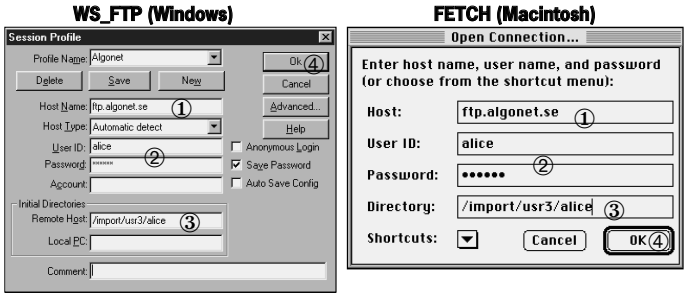
För att flytta filer från din egen dator till Internetleverantörens behöver du ett så kallat *FTP-program*. FTP står för File Transfer Protocol, och är en uppsättning regler för hur Internets datorer ska kommunicera då de skickar filer till varandra.

För Mac-ägare är shareware-programmet FETCH ett utmärkt val. Senaste versionen finns för nedladdning på följande adress: **http://fetchsoftworks.com/**och är knappt en megabyte stort.

Windowsanvändare har ett utmärkt FTP-program i ”WS_FTP” som kan hämtas i utvärderingsversion (30 dagar, efter det får man köpa licens). Det finns att hämta på adress: **http://www.ipswitch.com/** och är flera megabyte stort.

Viktigt.
I det här kapitlet utgår vi ifrån att du har en hemkatalog hos din Internetleverantör, och att du har rätt att ladda upp filer till den. Om du är osäker på hur det står till med den saken bör du fråga din leverantör. Om hemsidor inte tillåts bör du omedelbart säga upp ditt abonnemang och leta upp en annan leverantör. En bra sådan ger minst fyra megabyte utrymme för hemsidor utan extra kostnad.

Figur 12.1 a och b
Så här ställer du in FTP-programmen WS_FTP (vänster) och FETCH (höger) så att de kopplar upp mot din Internetleverantörs dator.



Att koppla upp FTP-programmet

Ladda ned lämpligt program och installera det. Koppla därefter åter upp dig mot din Internetleverantör och starta FTP-programmet.

Figur 12.1 a och b visar hur programmen ska konfigureras. Siffrorna nedan hänvisar till figurerna.

① Som host (”*värd”* på svenska) ska du ange din Internetleverantörs IP-adress. I exemplet är IP-adressen ”ftp.algonet.se”. Om du inte vet adressen så prova att sätta leverantörens namn mellan ”ftp.” och ”.se”. Om det inte fungerar är det bara att kontakta leverantören och fråga.

② I de här rutorna anger du ditt användarnamn och lösenord.

③ Som Internetabonnent har du en egen katalog på Internetleverantörens hårddisk. Den kallas hemkatalog och är döpt efter ditt användarnamn. Ibland måste ftp-programmet veta sökvägen till din hemkatalog för att hitta rätt när du kopplar upp dig. Alice hemkatalog har i exemplet sökvägen ”/import/usr3/alice”. Om du inte vet sökvägen till din hemkatalog kan du först prova att koppla upp utan att fylla i adressen. Om det inte fungerar måste du fråga din leverantör om rätt sökväg.

④ Till sist trycker du på OK-knappen, och programmet kopplar upp.

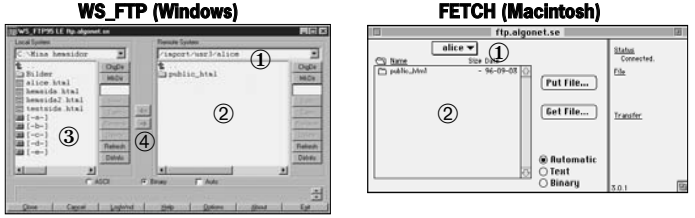
Att ladda upp filer

Om allt gått väl ska du nu befinna dig i din egen hemkatalog på Internetleverantörens dator, och fönstret bör likna figur 12.2 a eller b. Siffrorna nedan hänvisar till figurerna.

① Här ser du vilken katalog du befinner dig i. WS_FTP visar hela sökvägen, medan FETCH bara visar namnet på katalogen, som ju är samma som ditt användarnamn.

② I fönstret under katalognamnet visas innehållet. Med all sannolikhet finns här redan en katalog som heter ”public_html”. Det är i denna katalog alla dina hemsidor ska ligga. Om du inte ser någon public_html får du skapa den själv (se i marginalen på nästa sida).

Figur 12.2 a och b
Så här ser det ut när du är uppkopplad till Internetleverantörens dator.



Viktigt.
Om din hemkatalog inte innehåller underkatalogen ”public_html” måste du skapa den:
WS_FTP (Windows):
Klicka på knappen märkt ”MkDir” (*make directory*). Fyll i namnet **public_html** i rutan som kommer upp och klicka på OK. Eventuellt måste du trycka på knappen märkt ”Refresh” för att se den nya katalogen.
FETCH (Macintosh):
Gå in under ”Directories” i menyn och välj ”Create New Directory…”. Fyll i **public_html** i rutan som kommer upp och tryck på OK.

Uppladdningen sker på olika sätt i WS_FTP och FETCH:

WS_FTP (Windows):

Börja med att gå in i ”public_html” genom att dubbelklicka på katalogen i högerfönstret. ”Public_html” är antagligen tom, men ibland händer det att Internetleverantören lagt dit ett mindre HTML-dokument.

Titta nu på fönstret till vänster (③ i figur 12.2 a). Det visar din egen hårddisk. Leta dig fram till den katalog som innehåller dina hemsidor och öppna den. I figuren heter katalogen ”Mina hemsidor” och innehåller fyra HTML-dokument och en underkatalog för bilder.

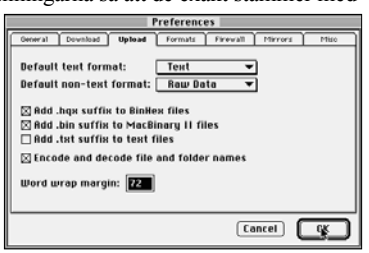
Markera alla dokument och underkataloger som du vill ladda upp (genom att hålla nere CTRL-tangenten kan du markera flera filer). Klicka sedan på högerpilen (④ i figuren). Filerna och katalogerna med innehåll laddas nu upp och visas slutligen i fönstret till höger.

FETCH (Macintosh):

Börja med att gå in i ”public_html” genom att dubbelklicka på katalogen. Den är antagligen tom, men ibland händer det att Internetleverantören lagt ett mindre HTML-dokument här.

Gå nu in under ”Customize” i menyn och välj ”Preferences”. Klicka på fliken märkt ”Upload”. Ändra inställningarna så att de exakt stämmer med de i figur 12.3. Klicka slutligen på OK.

Figur 12.3
Den här inställningen talar om hur FETCH ska ladda upp dina filer. Det är mycket viktigt att ”Add .txt suffix to textfiles” **inte** är ikryssad.



Nu kan du ladda upp filerna. Öppna menyn ”Remote” och välj ”Put folders and files…”. I rutan som dyker upp visar det övre fältet din hårddisk. Leta upp katalogen som innehåller dina hemsidor och öppna den. Markera alla filer och underkataloger som du vill ladda upp och klicka på ”Add”. Filerna visas då i det nedre fältet. Klicka till sist på ”Done” och uppladdningen börjar.

Att sätta rättigheter till filerna

När uppladdningen är färdig är det dags att provtitta på hemsidorna. Starta din webläsare och knappa in adressen till något av dina HTML-dokument. Följande formel gäller oftast för adressen:

http://www.*leverantör*.se/~*användarnamn*/*dokumentnamn.html*

Adressen till dokumentet ”hemsida.html” i exemplet blir alltså:

http://www.algonet.se/~alice/hemsida.html

Lägg märke till att katalogen ”public_html” aldrig ingår i adressen, trots att det ju egentligen är i den katalogen hemsidorna ligger.

I sällsynta fall kan webläsaren ge ett felmeddelande som ser ut ungefär så här när du försöker titta på dina nyuppladdade hemsidor:

Forbidden

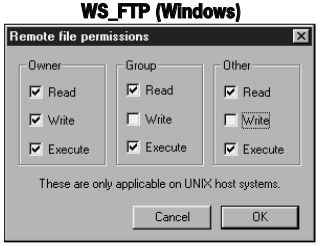
You don't have permission to access this URL on this server.

Det beror på att rättigheterna för filerna är felaktiga. Gör då så här: Koppla upp WS_FTP eller FETCH. Öppna ”public_html” och markera samtliga filer och kataloger som finns i den. I FETCH väljer du sedan ”Set Permissions” under menyalternativet ”Remote”. I WS_FTP markerar du filerna, klickar med högerknappen och väljer ”Chmod (UNIX)”.

Fönstret som dyker upp innehåller nio kryssrutor som anger vem som har rätt att läsa, skriva till och köra de filer du markerat. Kryssa i rutorna enligt figur 12.4 a eller b och tryck på OK. Hoppa därefter ett steg upp i katalogstrukturen så att du hamnar i hemkatalogen igen. Markera katalogen ”public_html” och upprepa proceduren så att den får samma rättigheter som filerna inuti. Prova nu åter att titta på dina hemsidor. Om det fortfarande inte fungerar bör du ringa leverantörens supportavdelning och be dem kontrollera att dina filer ligger på rätt ställe.

Tips! Om inget dokumentnamn anges i adressen letar webläsaren automatiskt efter ett dokument som heter *index.html*. Om du döper en av dina sidor, lämpligen öppningssidan, till index.html räcker det alltså att besökare vet ditt användarnamn och vilken leverantör du har för att de ska hitta fram till din hemsida.
Om Alice i exemplet haft en hemsida med namnet index.html hade adressen dit alltså blivit: http://www.algonet.se/~alice/

WS_FTP (Windows)



FETCH (Macintosh)

Figur 12.4 a och b
Kryssen talar om att alla ska få läsa dina hemsidor, men att bara du själv (Owner) ska ha rätt att göra ändringar i dokumenten (Write).

13 WEBBLÄSARENS DOLDA STÖD: CGI-script

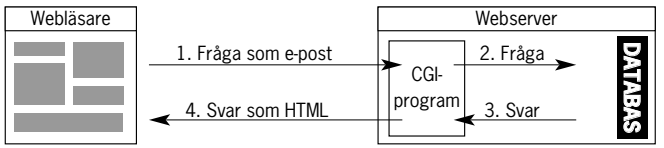
Kanske har du hört talas om CGI-script. Det är program som hjälper webläsaren att utföra sådana uppgifter som den normalt inte klarar, till exempel att söka i databaser, hantera formulär eller räkna antalet besökare på en hemsida. I det här kapitlet får du stifta bekantskap med CGI.

En stor svaghet med World Wide Web så som det såg ut i begynnelsen (vilket innebär för ett par-tre år sedan) var att hemsidorna var så odynamiska. En hemsida såg alltid likadan ut oavsett vem som laddade ned den, och en webläsare kunde bara passivt förmedla den färdiga informationen i HTML-dokumenten.

För att öka dynamiken skapades något som kallas Common Gateway Interface, förkortat CGI. CGI är en uppsättning regler för datorkommunikation som gör det möjligt för en särskild typ av datorprogram att kommunicera med en webserver, och därmed också med en webläsare. Programmen kallas för CGI-program, CGI-script eller ibland CGI-manus.

Tänk dig att en databas (till höger i figur 13.1) ska göras tillgänglig så att vem som helst kan söka i den via World Wide Web och en webläsare (till vänster i figuren). Normalt sett är detta inte möjligt. Webläsare klarar ju bara att visa hemsidor, inte att söka i databaser.

Men en webläsare kan skicka e-post. Därför skapas ett HTML-dokument med ett formulär (se kapitel 11) där besökaren får fylla i vad han



Figur 13.1
Webläsaren skickar en fråga till CGI-programmet, som söker i databasen. Svaret ges som ett HTML-dokument.

eller hon vill leta efter i databasen. Frågorna, eller sökorden, skickas därefter som e-post direkt till ett särskilt CGI-program på webservern (1 i figur 13.1). CGI-programmet är till skillnad från webläsaren särskilt utformat för att just söka i databaser. Den genomför sökningen (2) och får ett svar (3).

Att kunna ta emot data (i det här fallet frågor från ett formulär) i form av e-post från en webläsare är den ena finessen med CGI-program. Den andra är att de ofta kan skriva HTML-kod. CGI-programmet infogar alltså resultatet av databassökningen i ett HTML-dokument och sänder det till webläsaren (4), som slutligen visar det nyskapade dokumentet precis som om det vore en vanlig hemsida. Användaren märker oftast inte att ett CGI-script körts. Ändå är det processer liknande den här som sker varje gång du letar efter information på World Wide Web med hjälp av en **sökmotor**.

Sökmotorer. En sökmotor är ett program som automatiskt katalogiserar textinnehållet på WWWs hemsidor. Informationen organiseras sedan i en databas som du kan leta i med hjälp av en webläsare. Själva sökningarna i databasen sköts av ett CGI-program.
Alta Vista (www.digital.com) är en bra sökmotor.

Att använda egna CGI-program

Man kan använda flera olika programmeringsspråk för att skriva CGI-program. Vissa språk, som C och C++, måste *kompileras*. Det innebär att programmet skrivs som så kallad *källkod*, för att sedan med hjälp av en *kompilator* översättas till datorns eget språk.

Andra språk, till exempel PERL 4 och PERL 5, behöver inte kompileras. Ett program (eller *script* som okompilerade program ofta kallas) skrivet i PERL fungerar bara om servern där programmet körs är utrustad med en särskild tolk som rad för rad översätter koden till datorns eget språk. Detta sker varje gång programmet körs.

Eftersom kompilerade program inte behöver översättas då de körs, är de i allmänhet snabbare än icke-kompilerade program. PERL-script är å andra sidan lättare att jobba med, eftersom de ju inte behöver kompileras om varje gång man gjort en ändring.

För att du ska kunna använda egna CGI-program krävs dels själva programmet, dels att du har rätt att lägga program i katalogen ”**cgi-bin**” på Internetleverantörens webserver. Detta är långt ifrån säkert. CGI-program kan nämligen om de är felaktigt skrivna innebära en säkerhetsrisk. Börja alltså med att kolla om leverantören har några regler mot CGI-program.

Att hitta ett program är ganska enkelt; det vimlar av CGI-program på World Wide Web. En given startpunkt i letandet är Matt Wrights Script Archive, på adress http://worldwidemart.com/scripts/.

På bokens hemsida (du når den på adress http://www.etc.se/nyckeln/) finns mer information om hur CGI-script fungerar, hur du får in dem i dina HTML-dokument och hur koden justeras så att den fungerar på din webbserver.

cgi-bin. CGI-program ska normalt ligga i katalogen cgi-bin på webservern. Många Internetleverantörer är dock mycket försiktiga med vad de släpper in i denna katalog. Ett felaktigt skrivet program kan nämligen ställa till stora problem på servern.
Om du har tur är din leverantörs server utrustad med programmet CGIWrap. Då är det fritt fram att skapa en egen cgi-bin-katalog i ”public_html” och köra egna CGI-program därifrån. CGIWrap kontrollerar nämligen alla program och stoppar de som är skadliga. Mer information om CGIWrap hittar du på bokens hemsida.

14 Style sheets

FULL KONTROLL ÖVER SIDANS FORM:

Style sheets är ett kraftfullt komplement till HTML som låter dig formge hemsidor med samma precision som tidningar och andra trycksaker. I det här kapitlet tittar vi närmare på denna relativt nya teknik, som dramatiskt kommer att förändra villkoren för hur en hemsida kan se ut.

Ända sedan HTML-språket började användas till annat än publicering av forskningsrapporter har formgivare svurit över de begränsade möjligheterna till grafisk formgivning. Det har inte gått att använda grundläggande layouttekniker som att lägga text ovanpå annan text, styra textstorleken, finjustera textens placering på sidan eller ändra radavstånd. Inte ens en så grundläggande sak som indrag vid nytt stycke är det möjligt att åstadkomma i HTML utan att ta till specialtrick.

Idag använder formgivare därför ofta bilder för att få önskat utseende på rubriker, mellanrubriker och ibland till och med längre texter. Alla nät-

Figur 14.1
Med style sheets kan hemsidan för första gången få avancerad grafisk form. Samtidigt minskar nedladdningstiderna, eftersom bilder kan användas mer sällan. Allt du ser på hemsidorna till höger är vanlig text som formaterats med style sheets.

Viktigt.
I det här kapitlet kommer du att lära dig att koda style sheets på ett sådant sätt att dina sidor också ser bra ut i webbläsare som inte förstår style sheets.
Om du vill att dina besökare ska se dina vackra style sheet-formateringar är det dock viktigt att du rekommenderar Internet Explorer 4.0 eller Netscape Navigator 4.0.
Internet Explorer 3.0 har ett ofullständigt och bitvis felaktigt stöd för style sheets, och bör därför inte användas.

surfare vet resultatet: långa nedladdningstider och ibland svårlästa sidor. Den som någon gång försökt skriva ut sidor där texten egentligen är bilder vet också att det aldrig blir lyckat.

En ännu allvarligare konsekvens är att Netscape och Microsoft tävlar i att utöka själva HTML-språket med nya koder som ska öka layoutmöjligheterna. Många koder fungerar bara med en av läsarna (se kapitel 16 och 17), vilket strider mot World Wide Webs grundidé att HTML-språket ska vara en gemensam, öppen standard. Det finns till och med de som menar att kriget mellan Microsoft och Netscape riskerar att slita isär World Wide Web i två olika, webläsarspecifika delar.

För att hindra att detta sker har World Wide Web Consortium (W3C) arbetat fram en lösning som möjliggör avancerad layout utan att göra hemsidorna oläsliga för äldre webbläsare. Tekniken heter *Cascading Style Sheets* (CSS), eller bara ”style sheets”, och stöds när detta skrivs bara av Microsofts Internet Explorer 4.0 och Netscape Navigator 4.0.

Två olika typer av style sheets

HTML-koder är ganska fattiga. Med <h1> åstadkommer du till exempel bara en rubrik med största möjliga teckenstorlek. Men om du vill att rubriken ska vara grön, 30 pixlar hög, och visas med ett 10 pixlar stort indrag och typsnittet Verdana räcker inte <h1> långt.

Det är här style sheets kommer till användning. Tekniken låter dig helt enkelt tilldela befintliga koder, till exempel <h1>, alla de egenskaper som du tycker att de borde ha.

På det här sättet kan du definiera om samtliga HTML-koder som har en avslutningskod, till exempel <font>, <b>, <i> och <p>. Dessutom har det skapats två nya koder som bara används i samband med style sheets: och . Mer om detta längre fram.

Style sheets kan användas på två olika sätt: *lokalt* och *globalt*.

Den *lokala* metoden innebär att man definierar om enstaka HTML-koder på de ställen i dokumentet där det behövs. Då fungerar style sheets som en kraftfullare variant av <font>-koden.

Globala style sheets är kraftfullare. Med denna metod kan en eller flera koders innebörd ändras för hela HTML-dokumentet på en gång. Detta kan som vi snart ska se bespara dig massor av kodningsarbete.

I slutet av kapitlet kommer vi att titta på möjligheten att lagra ett globalt style sheet i ett separat dokument, och med hjälp av detta enda dokument styra utseendet på ett obegränsat antal hemsidor. Detta kallas ett *länkat style sheet*. Men först ska vi titta närmare på hur kodningen av lokala och globala style sheets fungerar i praktiken.

Lokala style sheets

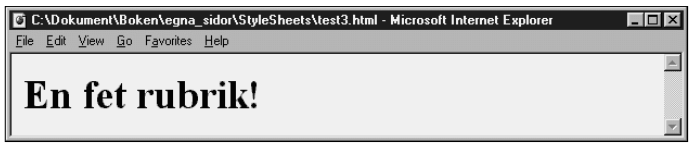
När style sheets används lokalt fungerar de ungefär som <font>-koden, med den skillnaden att du förutom textfärg, textstorlek och typsnitt även kan definiera saker som radavstånd, marginalbredd och indrag.

Exempel 14.2 a och figur 14.2 b visar en vanlig rubrik.

Exempel 14.2 a
En vanlig stor rubrik...

```
<h1>En fet rubrik!</h1>
```

Figur 14.2 b
... ser ut så här.



Nu ska vi definiera om <h1> med hjälp av ett lokalt style sheet. Det sker genom att vi lägger till det nya attributet style till <h1>.

Exempel 14.3 a visar den nya koden, och figur 14.3 b vår nya rubrik.

Exempel 14.3 a
Här definieras <h1> om med hjälp av ett style sheet.

```
<h1 style="text-indent:50px; font-size:30px; font-family:Impact,
Helvetica">En fet rubrik!</h1>
```

Figur 14.3 b
Resultatet: rubriken visas med typsnittet Impact, och med ett 50 pixlar stort indrag. Om webbläsaren varit av äldre modell och inte klarat style sheets hade rubriken visats som en vanlig <h1>-rubrik.

I exempel 14.3 a anger vi att rubriken ska visas med ett 50 pixlar stort indrag (Text-indent:50px), en textstorlek på 30 pixlar (Font-size:30px) och antingen typsnittet Impact eller Helvetica (Font-family:Impact, Helvetica). Som framgår av exemplet radas attributen till style upp enligt den här mallen:

style=”*attribut1*: *värde1; attribut2: värde2;* (osv)”

Lägg märke till att hela raden av attribut och värden måste stå inom citationstecken och att attributen separeras med semikolon.

Tips!
I bilaga 2 hittar du en lista över attribut som kan användas tillsammans med style sheets.

Det finns många fler style-attribut som styr textens egenskaper. I bilaga 2 hittar du en lista över attribut, men här är några särskilt viktiga. Experimentera gärna med att lägga fler attribut till exempel 14.3 a.

font-style: *normal* eller ***italic***. Skapar normal respektive kursiv text.

font-weight: *normal* eller ***bold***. Skapar normal respektive fet text.

font-size: *värde* anger storlek på text. Kan anges i pixlar (px), punkter (pt), centimeter (cm), tecknets “boxhöjd” (em) eller inch (in).

font-family: alternativ1, alternativ2, alternativ3 (osv). Anger vilket

Tips!
En svaghet med style sheets är att de liksom <font face=*typsnitt*> kräver att användaren har särskilda typsnitt installerade. Annars ser inte sidan ut som planerat.
På adress http://www.microsoft.com/msdownload/ finns ett gratis typsnittspaket för både PC och Mac. En lösning är att bara använda dessa typsnitt i dina style sheets, och att uppmana alla besökare att också ladda ned paketet.
Typsnitten som ingår är: Arial, Arial Black, Comic Sans Ms, Courier New, Impact, Times New Roman och Verdana. På sidan 31 kan du se hur typsnitten ser ut.
Läs också avsnittet om dynamiska typsnitt, sist i det här kapitlet.

typsnitt texten ska visas med. Om du som i exempel 14.3 a anger mer än ett typsnittsnamn kommer webbläsaren att i tur och ordning försöka hitta typsnitten i användarens dator. Om inte det första finns försöker datorn hitta det andra i listan och så vidare. Om inget av typsnitten finns visas texten med webbläsarens standardtypsnitt. Förutom typsnittsnamn kan du ange en typsnittsgrupp. Grupperna är: *sans-serif, serif, cursive, fantasy* och *monospace.* Det är bra att alltid avsluta typsnittslistan med en typsnittsgrupp, till exempel ”Font-family: Verdana, Helvetica, sans-serif”. Då minskar risken för att datorn måste falla tillbaka på standardtypsnittet.

Globala style sheets

Nackdelen med lokala style sheets, till exempel det vi använde i exempel 14.3 a, är att de bara kan användas en gång. Om vi hade velat ha en likadan rubrik längre ned i dokumentet hade vi alltså blivit tvungna att definiera om typsnitt, textstorlek och så vidare med hjälp av style-attributet. Om texten varit lång och innehållit många rubriker hade detta inneburit massor av arbete. Därför finns globala style sheets.

Den stora skillnaden är att man med ett globalt style sheet en gång för alla kan definiera hur text och rubriker ska se ut, och sedan använder samma definition i hela HTML-dokumentet. En viktig finess är att de definierade elementen ”ärver” varandras egenskaper enligt fastställda regler. Detta besparar, som vi snart ska se, HTML-kodaren yttterligare slit.

Det finns tre olika metoder att definiera globala style sheets. Vi ska gå igenom dem var för sig.

Typografi på World Wide Web

Textstorlek och radavstånd
Textens storlek, eller *grad* som det också kallas, mäts från den lägsta punkten till den högsta. Radavståndet är utrymmet mellan radernas *baslinjer*.

Måttenheter
Textstorlek och radavstånd kan i style sheets anges som punkter (pt), inch (in), centimeter (cm) eller pixlar (px).
1 punkt = 0,352 mm
1 inch = 25,4 mm

Serif och sans-serif
Det finns två olika huvudgrupper av typsnitt: *serif* och *sans-serif.* Det som skiljer dem åt är att serifer har fötter och flaggor, något sans-seriferna saknar (se figuren nedan).
Flaggorna och fötterna underlättar för ögat att följa texten. Seriferna passar därför bäst för längre texter, medan sans-serifer lämpar sig bäst för rubriker. Arial, Impact och Helvetica är sans-serifer. Times är en serif.

Typ 1: Omdefinierade HTML-koder

Alla globala style sheets skapas genom att den nya koden <style>, med avslutningskod </style>, placeras i HTML-dokumentets <head>-sektion. Mellan <style> och </style> definierar vi sedan vårt globala style sheet. Exempel 14.4 a visar hur det går till och i figur 14.4 b ser du resultatet. Nedan går vi igenom de viktigaste raderna i koden.

Exempel 14.4 a
Här definierar vi om HTML-koderna <p> och . De nya egenskaperna kommer att gälla varje gång koderna används i dokumentet och tekniken kallas därför ”globalt style sheet”.

```
1  <html><head><title>Titel</title>
2  <style type="text/css">
3  <!- -
4  p {font-family:Arial, helvetica; color:#333333}
5  b {text-transform:uppercase; color:#000000}
6  - ->
7  </style>
8  </head><body>
9  <p>Här är en text med ett <b>stort </b>ord i.</p>
10 Detta är vanlig text.
11 </body></html>
```

Figur 14.4 b
Resultatet: <p> och har fått helt nya egenskaper.

De viktigaste raderna i exempel 14.4 a

1–2. <style>-koden ska placeras mellan <head> och </head>. Attributet type=”text/css” måste alltid vara med. Det talar om för webbläsaren att det som följer är definitionen av ett style sheet.

3. För att inte webbläsare som inte förstår style sheets ska bli förvirrade har hela definitionen lagts mellan kommentartecken (<!- - och - ->).

4. Koden <p> har vi hittills bara använt för att göra nytt stycke med. Nu tilldelar vi den två nya egenskaper. Dels ska den visa text med typsnittet Arial eller Helvetica, dels ska texten visas med grå färg (#333333). Lägg märke till att definitionen ska stå mellan vågparenteser ({ och }).

5. Här definierar vi om koden för fet text, . ”Text-transform:uppercase” betyder att texter som omges av och ska visas med versaler (stora bokstäver). Vi tala också om att texten ska vara svart (#000000).

7–8. När de två HTML-koderna är omdefinierade avslutas definitionen med </style> och därefter kan <head>-sektionen också avslutas.

9. I normala fall hade bara skapat ett nytt stycke. Men eftersom vi definierat om blir resultatet nu att den efterföljande texten visas enligt specifikationen på rad 4. Ett av orden i texten omgas av och . Det innebär att denna del visas med fet stil som vanligt, men dessutom att bokstäverna visas med svarta versaler. Sist på rad 9 avslutas stycket med . Det betyder att specifikationen från rad 4 inte längre gäller och den efterföljande texten visas med standardtypsnitt.

Om dokumentet hade varit längre hade de omdefinierade och kunnat användas flera gånger.

Typ 2: Klasser

Om man vill definiera många olika texttyper är det bättre att skapa så kallade ”klasser” än att som i exempel 14.4 a definiera om en hel HTML-kod.

Exempel 14.5 a visar hur detta går till. Resultatet ser du i figur 14.5 b.

Exempel 14.5 a
Här tilldelas och varsin klass: ”special” respektive ”stor”.

```
1  <head><style type="text/css">
2  <!--
3  p.special {font-family:Arial, helvetica; color:#333333}
4  b.stor {text-transform:uppercase; color:#000000;
5  background:#FFFF00}
6  -->
7  </style></head><body>
8  <p class="special">Det här är en speciell text med <b
9  class="stor">stora ord</b> i.</p>
10 Detta är vanlig text
```

Figur 14.5 b
Med attributet class=”klassnamn” ges avsnitt i texten särskilda egenskaper, till exempel en särskild bakgrundsfärg.

De viktigaste raderna i exempel 14.5 a

3. Den stora nyheten är att vi här inte definierar om själva -koden som i exempel 14.4 a, utan skapar en ”klass” till . Namnet på klassen är ”special”. Klassnamnet ska föregås av en punkt och placeras direkt efter namnet på koden som klassen hör till, i det här fallet P.

4–5. På raden under skapar vi en klass till som vi döper till ”stor”.

8. Genom att lägga attributet class=”special” till talar vi om att den efterföljande texten ska visas enligt -klassen ”special”. På samma sätt träder -klassen ”stor” i funktion när vi skriver .

I exemplet knöt vi de båda klasserna ”special” och ”stor” till HTML-koderna och . Det är också möjligt att skapa klasser som inte är kopplade till någon särskild HTML-kod. Allt man behöver göra är att som i exempel 14.6 (nedan) utelämna kodnamnet då klassen definieras. Punkten måste dock vara kvar. Jämför gärna med exempel 14.5 a.

Exempel 14.6
Klassen special är inte bunden till någon särskild HTML-kod. Lägg märke till att punkten måste vara kvar före klass-namnet.

```
<style type="text/css">
<!--
.special {text-decoration:underline}
-->
</style>
```

Klassen ”special” kan nu användas tillsammans med vilket HTML-kod som helst i dokumentet. Exempel 14.7 a visar hur det skulle kunna se ut och figur 14.7 b resultatet i webbläsaren.

Exempel 14.7 a och figur 14.7 b
Med attributet class=”special” kan klassen ”special” appliceras på vilka koder som helst.

```
Detta är en text som <font class="special">använder</font> klassen <b class="special">Special</b> både här och där.
```

Lägg märke till att klassen ”special” nu används både tillsammans med <font> och . Notera också att fortfarande gör texten fet. Att med hjälp av attributet class=”special” tilldelas klassen ”specials” egenskaper betyder alltså inte att dess inbyggda funktion – att göra text fet – går förlorad.

Typ 3: Individuella style sheets (id)

Den tredje typen av style sheet heter ID och fungerar i princip som den icke-kodbundna klassen i exempel 14.7 a. Det finns dock ett par viktiga skillnader som vi snart ska titta närmare på.

En ID är inte bunden till någon HTML-kod. Den definieras genom att namnet föregås av ett brädgårdstecken (#). I övrigt fungerar definitionen av en ID precis som definitionen av en klass.

Exempel 14.8 a visar hur ID används. Figur 14.8 b visar resultatet.

Exempel 14.8 a
I det här exemplet skapar vi en ID med namnet ”utrop”. Lägg märke till att namnet måste inledas med ett brädgårdstecken (#).

```
<style type=”text/css”>
<!–
#utrop {font-size:40px}
–>
</style></head><body>
Det här är en text med ett <font id=”utrop”>utrop</font> i.
```

Figur 14.8 b
När webbläsaren visar koden ser vi att en ID fungerar precis som en klass med den skillnaden att vi nu använder attributet id=”namn”.

Som framgår av exempel 14.8 a fungerar en ID precis som en klass. Enda synliga skillnaden är att en id aldrig kan kopplas till en särskild HTML-kod (som vi gjorde med en klass i exempel 14.5 a), att namnet inleds med ett brädgårdstecken och att id appliceras på HTML-koden med attributet id=”*namn*”.

Det finns dock viktiga skillnader mellan klasser och id, och de blir uppenbara när man försöker kombinera id med klasser, lokala style sheets och vanlig HTML. Det ska vi göra nu.

Alltihop på en gång

Det finns sammanlagt fem olika sätt att ändra egenskaper hos en text: med vanlig HTML (till exempel med <font>-koden från kapitel 3), med globalt omdefinierade HTML-koder (exempel 14.4), med klasser (exempel 14.5), med id (exempel 14.8) och med lokala style sheets (exempel 14.3).

Om man vill att en sida ska se bra ut både med webbläsare som klarar style sheets (när detta skrivs bara Netscape Navigator 4.0 och Internet Explorer 4.0) och äldre webbläsare måste man kombinera dessa fem metoder. För att detta ska fungera utan konflikter innehåller style sheet-tekniken en inbördes ”hackordning”. Vissa formateringssätt väger tyngre än andra och har därmed sista ordet vid eventuella konflikter.

Figur 14.9 visar denna hackordning, från tyngst till lättast.

Figur 14.9
Hackordningen avgör vilken typ av formatering som ska få sista ordet när flera olika typer krockar med varandra.

Tyngst	Lokalt style sheet
↓	Global id
	Global klass
	Globalt omdefinierad html
Lättast	Vanlig html

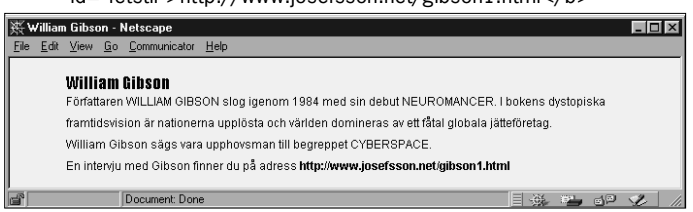
För att demonstrera hur hackordningen fungerar ska vi använda alla de olika metoderna för textformatering på en gång, och sedan se resultatet i två olika webbläsare.

Exempel 14.10 a visar koden. Standardkoder som <html> och <head> har tagits bort av utrymmesskäl, men ska naturligtvis vara med när du själv experimenterar. Figur 14.10 b och c visar resultatet i Netscape 4.0 (som förstår style sheets) respektive Netscape 3.0 (som inte förstår style sheets). På nästa sida går vi igenom de viktigaste koderna.

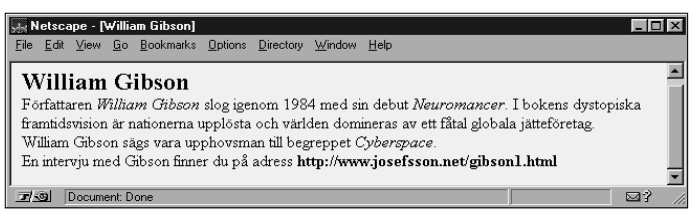
Exempel 14.10 a
I det här exemplet används alla metoderna för textformatering på en gång. Sidan kan därför ses även med webbläsare som inte förstår style sheets.

```
1  <style type="text/css">
2  body {font-family:arial, helvetica, sans-serif; Margin-left:50px;
3  font-size:12px; Line-height:22px;}
4  .rubrik {font-size:18px; font-family:Impact}
5  b {font-weight:normal}
6  #fetstil {font-weight:bold}
7  i {font-style:normal; text-transform:uppercase}
8  </style>
9  <body>
10 <font size=”6” class=”rubrik”><b>William Gibson</b></font>
11 <br />
12 Författaren <i>William Gibson</i> slog igenom 1984 med sin
13 debut <i>Neuromancer</i>. I bokens dystopiska framtidsvision är
14 nationerna upplösta och världen domineras av ett fåtal globala jät-
15 teföretag.<br />William Gibson sägs vara upphovsman till begrep-
16 pet <i>Cyberspace</i>.<br />En intervju med Gibson finner du
17 på adress <b
   id=”fetstil”>http://www.josefsson.net/gibson1.html</b>
```

Figur 14.10 b
Så här ser sidan ut sedd med Netscape Navigator 4.0. Alla style sheet-formateringarna fungerar fullt ut.



Figur 14.10 c
Så här blir sidan när den ses med Netscape Navigator 3.0, som inte förstår style sheets. Enklare, men fortfarande fullt läsbart.



De viktigaste raderna i exempel 14.10 a

2. Här tilldelas <body>-koden nya egenskaper. Eftersom <body> omsluter hela hemsidan kommer all text att utgå från denna specifikation: typsnitt sans-serif, 50 pixlars marginal, en textstorlek på 12 pixlar och ett radavstånd (line-height) på 22 pixlar.

4. Här skapar vi klassen ”rubrik”. Den är 18 pixlar hög och ska visas med typsnittet Impact.

5. Attributet font-weight (ungefär *typsnitts-vikt*) har två möjliga värden: normal eller bold (fet). På den här raden ger vi koden en normal font-weight, vilket i praktiken innebär att vi tar ifrån dess funktion, som ju normalt är att göra text fet (bold). Poängen med detta kommer strax att framgå.

6. En id med namnet ”fetstil” skapas.

7. Attributet font-style har två möjliga värden: normal och italic (*kursiv*). Vi bestämmer på denna rad att inte längre ska ge kursiv text. Istället får egenskapen att visa text med versaler (stora bokstäver).

10. Texten inleds med en rubrik som formateras med hjälp av <font>-koden. Här uppstår en konflikt. Å ena sidan säger attributet size=”6” att rubriken ska visas med det näst största typsnittet. Å andra sidan säger attributet class=”rubrik” att rubriken ska visas enligt definitionen på rad 4. Det är nu hackordningen i figur 14.9 kommer till användning. Enligt den väger klasser tyngre än vanlig HTML. Alltså kommer webbläsare som förstår style sheets att ignorera size=”6” och istället visa rubriken enligt klassdefinitionen. För webbläsare som inte förstår style sheets behövs ingen hackordning; de förstår överhuvudtaget inte vad class=”rubrik” betyder och använder därför det välbekanta size=”6”.

Rubriken omges dessutom av och . För webbläsare som förstår style sheets har inte längre någon effekt, vi neutraliserade ju på rad 5. För äldre webbläsare fungerar precis som vanligt.

I figur 14.10 b och c kan du se hur rubriken ter sig i de olika webläsarna. Netscape 4.0 visar en 12 punkters rubrik med typsnittet Impact. Netscape 3.0:s rubrik är av storlek 6, fetad och visas med standardtypsnitt.

12–16. Vissa ord i texten omges av koderna för kursiv stil, och . Som framgår av figurerna 14.10 b och c reagerar de två webläsarna olika. Netscape 4.0 går efter definitionen på rad 7 och visar texten med versaler, men utan kursiv stil. Netscape 3.0 visar den kursivt som vanligt.

17. Texten avslutas med en adress som vi vill ska visas med fet stil i båda webläsarna. Men hur ska det gå till? Efter omdefinitionen på rad 5 betyder ju inte längre något i Netscape 4.0. Lösningen ligger i det id vi skapade på rad 6. Med attributet id=”fetstil” ger vi helt enkelt tillfälligt tillbaka dess egenskaper. Och eftersom en id väger tyngre än en omdefinierad HTML-kod, kommer adressen att visas med fet stil även i Netscape 4.0, vilket också framgår av figur 14.10 b.

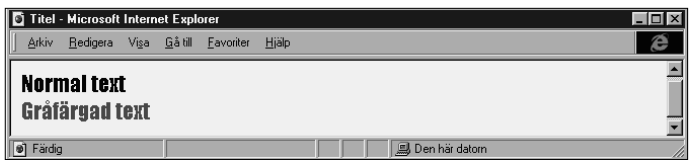
Att kombinera klasser och ID

Eftersom en ID väger tyngre än en klass i hackordningen går det bra att förse en HTML-kod med både klass och ID. Hur detta fungerar ser du i exempel 14.11 a och figur 14.11 b.

Exempel 14.11 a
Klassen ”normal” och ID ”special” innehåller olika textfärg. På sista raden kombineras de i en och samma <font>-kod.

```
<style type=”text/css”>
.normal {color:#000000; font-family:impact; font-size:22px}
#special {color:#666666}
</style>
<font class=”normal”>Normal text</font><br />
<font class=”normal” id=”special”>Gråfärgad text</font>
```

Figur 14.11 b
Resultatet. Eftersom en id väger tyngre än en klass blir det id=”special” som får ange färgen, och texten visas grå.



På sista raden kombineras class=”normal”, enligt vilken texten ska vara svart, med id=”special”, som säger att texten ska vara grå. Eftersom id enligt figur 14.9 har större tyngd än en klass, blir det id som får sista ordet och texten blir grå.

Du kan bara lägga in en class=”*namn”* och en id=”*namn”* i varje HTML-kod. Om du skulle råka tilldela en kod två klasser eller id kommer webbläsaren att använda den första, men ignorera den andra. I exempel 14.12 skulle alltså klassen ”normal” användas, och ”fet” ignoreras.

Exempel 14.12
Varje HTML-kod får bara innehålla en klass och en id. Här ignoreras class=”fet”.

```
<font class=”normal” class=”fet”>Normal text</font>
```

Blockelementens hemlighet

De enda koder som inte kan definieras om med hjälp av lokala eller globala style sheets är sådana som saknar avslutningskod, till exempel <br />. De övriga delas in i två grupper:

Första gruppen innehåller alla koder som liksom <font>, och kan sprängas in var som helst i till exempel en textrad utan att någon ny rad behöver skapas.

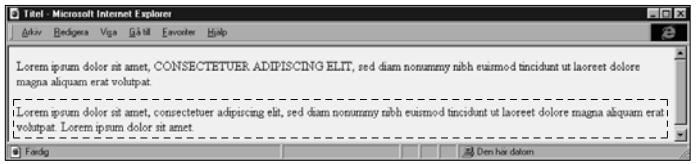
Till andra gruppen hör koderna som automatiskt skapar en ny rad då de inleds och avslutas. Exempel på sådana är <h1>, <p> och <blockquote>. Koderna i denna grupp brukar kallas *blockelement,* eftersom de i praktiken

skapar ett rektangulärt block, eller avsnitt, i texten. I exempel 14.13 a visas skillnaden mellan vanliga HTML-element och blockelement.

Exempel 14.13 a
En vanlig kod som <font> kan sprängas in var som helst i en text.

```
Lorem ipsum dolor sit amet, <font style="text-transform:uppercase">consectetuer adipiscing elit</font>sed (text kortad) erat volutpat.<p>Lorem (text kortad) amet.</p>
```

Figur 14.13 b
Ett blockelement, till exempel , börjar och slutar med en radbrytning. Den skapar därmed ett rektangulärt block, som i figuren markerats med en streckad linje.



Lägg märke till att avsnittet som formateras med hjälp av <font> börjar och slutar mitt i en rad. Avsnittet som omges av ... får däremot automatiskt en radbrytning före och efter. skapar därmed ett block som i figur 14.13 b är markerat med en streckad linje.

Med hjälp av en särskild uppsättning style sheet-attribut kan detta block förses med allt från ram till egen bakgrundsfärg. Du hittar en lista över dessa särskilda blockattribut i bilaga 2. I exempel 14.14 a använder vi några av dem för att skapa en svart ruta med vit text. Jämför gärna med exempel 14.13 a.

Exempel 14.14 a
Med de särskilda blockattributen kan man ge blocken som skapats med till exempel allt från ram till bakgrundsfärg.

```
<style type="text/css">
.faktaruta {border-style:solid; border-width:2px; padding:5px; background-color:#000000; color:#FFFFFF; margin-top:10px; font-weight:bold}
</style>
<body>
Lorem ipsum dolor sit amet, <font style="text-transform:uppercase">consectetuer adipiscing elit</font>sed (text kortad) erat volutpat.
<p class="faktaruta">Lorem (text kortad) amet.</p>
```

Exempel 14.14 b
Resultatet i Netscape Navigator 4.0. Hela blocket visas med svart bakgrundsfärg.

och

Hittills har vi med hjälp av style sheets kompletterat vanliga HTML-koder med nya attribut och egenskaper. Men det finns också två speciella koder som ofta används tillsammans med style sheets. Den första heter .. , och den kanske du känner igen från sidan 32. Den andra är helt ny och heter .... Gemensamt för dem båda är att de utan attribut helt saknar uppgift. Detta gör dem särskilt lämpade att tilldelas uppgifter med hjälp av style sheets. Av de två koderna är ett blockelement, medan är ett vanligt icke-blockskapande element. I exempel 14.13 a hade vi alltså kunnat byta ut mot och <font> mot och fått oförändrat resultat. Tänk bara på att äldre webbläsare helt ignorerar och . Om vi hade bytt ut alla <font> och <b> i exempel 14.10 a mot , hade resultatet alltså blivit att en äldre webbläsare som Netscape Navigator 3.0 (figur 14.10 c) inte visat några textformateringar överhuvudtaget.

Bild och text i lager

Möjligheten att lägga bild och text ovanpå varandra är en av de kraftfullaste finesserna med style sheets. Genom att arbeta i lager kan man skapa allt från skuggeffekter till fullfjädrad tidningslayout på sin hemsida.

Vi ska nu se hur det går till att skapa en rubrik med skugga. Först definierar vi klassen ”Rubbe”, som bland annat innehåller rubrikens typsnitt och färg. Dessutom skapar vi en id, ”Ljusare”, som ger en ljusgrå färg.

Exempel 14.15 a
En klass och en id definieras.

```
<style>
.Rubbe {font-family:Impact; font-size:50px; font-weight:bold}
#Ljusare {color:#CCCCCC}
</style>
```

Nästa steg blir att få webbläsaren att visa rubriken på en exakt position. För detta använder vi ett lokalt style sheet (bläddra tillbaka till exempel 14.3 a om du glömt hur ett lokat style sheet fungerar):

Exempel 14.15 b
Med hjälp av ett lokalt style sheet positionerar vi rubriken exakt där vi vill ha den.

```
<div class="Rubbe style="position:absolute; left:50px; top:5px" >Style sheets i lager!</div>
```

Lägg märke till att vi använder -koden. Bara blockobjekt kan läggas ovanpå varandra. <h1> och hade alltså också gått bra.

Position:absolute betyder att positionsbestämningen gäller från sidans vänster- respektive överkant. Left:50px och top:5px anger hur långt från vänster- och överkanten blocket ska ligga.

Figur 14.15 c
Resultatet av kodningen i exempel 14.15 a och b. En exakt placerad rubrik.

Resultatet av kodningen ser du i figur 14.15 c ovan. En fet rubrik i typsnittet Impact som visas 50 pixlar från den vänstra och 5 pixlar från den övre fönsterkanten.

Nästa moment blir att lägga ytterligare en rubrik ovanpå den första. Därför lägger vi till följande kodrad:

Exempel 14.15 d
Genom att lägga till ytterligare en rubrik med en liten förskjutning skapas skuggeffekten.

```
<div class="Rubbe" id="Ljusare" style="position:absolute;
left:48px; top:3px">Style sheets i lager!</div>
```

Lägg märke till att vi gör den här versionen av rubriken ljusare med hjälp av id ”Ljusare”. Notera också att rubriken placeras 2 pixlar närmare vänster- och överkanten. Resultatet av detta ser du i figur 14.15 e.

Figur 14.15 e
Slutresultat: en rubrik med skugga.

Styr dina lager med z-index

Att den gråa rubriken i figur 14.15 e lade sig ovanpå den svarta, och inte tvärtom, beror helt enkelt på att den svarta rubriken kom först i HTML-dokumentet. Webbläsaren placerar alltid de första objekten underst.

Här ser du hur webbläsaren organiserar lager:

Exempel 14.16
Det första objektet hamnar alltid underst.

```
<div class="skugga>Jag hamnar underst</div>
<div class="rubrik>Jag hamnar i mitten</div>
<div class="dekoration>Jag hamnar överst</div>
```

Ofta behöver man ett mer praktiskt sätt att hålla reda på lagerordningen. Därför finns attributet z-index. Ju högre värdet på z-index är, desto högre upp hamnar lagret. Om vi vill lägga den svarta rubriken ovanpå den grå i exempel 14.15 d, ska z-index alltså anges som i exempel 14.17.

Exempel 14.17
Med attributet z-index kan lagrens placering styras exakt.

```
<div class="Rubbe ID=Ljusare style="position:absolute; left:48px;
top:3px; z-index:1">Style sheets i lager!</div>
<div class="Rubbe style="position:absolute; left:50px; top:5px; z-
index:2">Style sheets i lager!</div>
```


Länkade style sheets

En finess som kan bespara dig väldigt mycket arbete är möjligheten att lägga själva style sheet-definitionen i ett separat dokument och sedan anropa detta från ett obegränsat antal HTML-dokument som behöver använda definitionen.

Om du till exempel vill utnyttja definitionerna i exempel 14.10 a i mer än ett dokument klipper du ut allt mellan <style> och </style> (rad 2–7) och klistrar in det i ett tomt textdokument. Exempel 14.18 visar hur det nya dokumentet ska se ut. Observera att det *bara* är själva definitionerna som får vara med, inga HTML-koder.

Exempel 14.18
En definitionsfil innehåller bara själva style sheet-definitionerna. Inga HTML-koder får finnas med.

```
body      {font-family:arial, helvetica, sans-serif; margin-left:50px;
          font-size:12px; line-height:22px;}
.rubrik {font-size:18px; font-family:Impact}
b {font-weight:normal}
#fetstil {font-weight:bold}
i {font-style:normal; text-transform:uppercase}
```

Spara dokumentet med ett unikt namn följt av suffixet ”.css” (*cascading style sheet*), till exempel ”min_stil.css”.

Inledningen till HTML-dokumenten som ska använda definitionerna i ”min_stil.css” ska sedan se ut som i exempel 14.19.

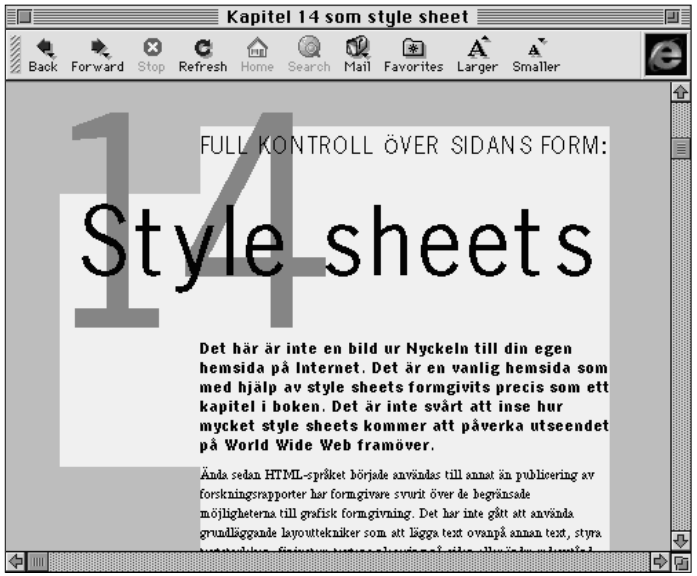
Exempel 14.19
Med hjälp av koden <link /> laddas de separat lagrade definitionerna in. Observera att <link /> avslutas “inuti” taggen och ska ligga i dokumentets <head>-sektion.

```
<html>
<head><title>Titel</title>
<link rel="stylesheet" type="text/css" href="min_stil.css" />
</head>
<body> (resten av dokumentet)
```

Den nya koden <link /> (länk) ska alltså placeras i dokumentets <head>-sektion. Attributen rel=”stylesheet” och type=”text/css” talar om att länken leder till ett style sheet. href=”min_stil.css” anger slutligen namnet på vårt länkade style sheet. I exemplet är adressen relativ, men även absoluta sökvägar går bra (till exempel ”http://www.etc.se/stilar/min_stil.css”). Lägg märke till att <link /> avslutas i slutet av taggen.

Sedan är det fritt fram att i resten av dokumentet använda de klasser, id och omdefinitioner som ligger i ”min_stil.css”. Därmed kan du också på några sekunder ändra utseende på rubriker, brödtext och mycket annat i hundratals dokument. Gå bara in och ändra i ”min_stil.css” och dina ändringar slår igenom i samtliga HTML-dokument som är länkade dit.

Figur 14.20
Style sheet-tekniken har potential att äntligen öppna Internet för det 500-åriga typografiska kunnade som används när man gör böcker och tidningar. Figuren visar en hemsida som med hjälp av style sheets formgivits precis som inledningen till det här kapitlet.



Dynamiska typsnitt

Genom att kombinera alla de olika style sheet-funktioner som vi gått igenom i det här kapitlet kan man formge hemsidor precis som en tidning eller en bok. Hemsidan i figur 14.20 innehåller en mängd lager, klasser och ID. Resultatet blir att den nästan exakt ser ut som inledningen till det här kapitlet.

Det stora problemet är typsnitten. För att sidan i figur 14.18 ska visas på rätt sätt måste den som ser den ha typsnitten Times och News Gothic installerade i sin dator. Times är inget problem, det finns installerat när man köper sin Macintosh eller PC. Men med News Gothic är det värre. Det är ett kommersiellt typsnitt som bara formgivare normalt har i sina datorer.

Lösningen heter *dynamiska typsnitt* och går ut på att webbläsaren automatiskt från Internet laddar ned de typsnitt som ingår i hemsidan. Därmed kan formgivaren skapa sidor utan att behöva bekymra sig om vilka typsnitt slutanvändaren har.

Både Netscape Navigator 4.0 och Internet Explorer 4.0 stöder dynamiska typsnitt. Men som så många gånger förr har Netscape och Microsoft gjort livet svårt för sina kunder genom att välja två helt olika system.

Tips!
I avsnittet om style sheets på bokens hemsida kan du läsa mer om dynamiska typsnitt. Adressen är: www.etc.se/nyckeln/

Netscapes dynamiska typsnitt

Netscapes variant av dynamiska typsnitt skapas med hjälp av programmet Typograph 2.0 från företaget HexWeb. Programmet tar ett typsnitt (eller *font* som det ofta kallas) som finns installerat i datorn, till exempel News Gothic, och skapar en *typsnittsdefinition*. Filen blir bara runt 15 KB stor och namnet ska sluta på *.pfr*. Definitionsfilen läggs på webservern och anropas sedan från HTML-dokumentet med hjälp av <link>-koden. Exempel 14.21 visar hur det ska se ut.

Exempel 14.21
Netscape använder LINK-koden för att ladda ned dynamiska typsnitt. Här heter typsnittsdefinitionen *newsgothic.pfr*.

```
<head>
<link rel="fontdef" src="newsgothic.pfr" />
<style>
.rubrik {font-size:18px; font-family:News Gothic}
</style>
</head>
<body> <h1 class="rubrik>Rubrik i News Gothic!</h1>
```

Lägg märke till att <link> ska ligga i <head>-sektionen och saknar avslutningskod.

Microsofts dynamiska typsnitt

Microsofts version av dynamiska typsnitt skapas på ungefär samma sätt som Netscapes, men med företagets eget program WEFT (*web embedding font tool*). Namnet på definitionsfilen har förlängningen *.eot*.

Definitionsfilen läggs på webservern och anropas sedan med style sheet-kommandot @font-face. Om definitionsfilen heter ”newsgothic.eot” ska koden se ut så här:

Exempel 14.22
Explorer använder <style>-attributet @font-face för att ladda in den dynamiska fonten.

```
<head>
<style>
@font-face {font-family:News Gothic; src:newsgothic.eot}
.rubrik {font-size:18px; font-family:News Gothic}
</style>
</head>
<body> <h1 class="rubrik>Rubrik i News Gothic!</h1>
```

På bokens hemsida hittar du mer information om hur det går till att skapa dynamiska typsnitt och hur Netscapes och Internet Explorers versioner kan kombineras så att dina sidor fungerar med båda läsarna.

I bilaga 2 hittar du listor över de attribut som används tillsammans med style sheets.

Viktigt.
Koderna i detta kapitel fungerar bara med Internet Explorer 4.0 och Netscape Navigator 4.0. Bilaga 2 innehåller en förteckning över attribut som kan användas tillsammans med style sheets.

Koderna i sammanfattning

Nya HTML-koder

<style>...</style>
Skapar ett globalt style sheet. Måste placeras i <head>-sektionen.

Attribut till <style>

type="*text/css*"
Talar om att koden som följer är en cascading style sheet (CSS).

<link />
Anropar ett länkat style sheet eller en dynamisk font (endast Netscape Navigator 4.0). Placeras i HEAD-delen och saknar avslutningskod.

Attribut till <link />

rel="*stylesheet*" eller "*fontdef*"
Anger om länken leder till ett länkat style sheet eller till en dynamisk font.
type="*text/css*"
Anger MIME-typ vid länk till style sheet.
src="*adress*"
Anger adress till dynamisk font.

...
Saknar egenskaper. Måste tilldelas egenskaper med class eller id.

...
Samma som , men blockelement.

Nya ATTRIBUT TILL HTML-koder

style="*attribut1: värde1; attribut2: värde2;* (osv)"
Skapar ett lokalt style sheet. Kan användas tillsammans med alla HTML-koder som har avslutningskod.
class="*klassnamn*"
Knyter en HTML-kod till en fördefinierad klass.
id="*namn*"
Knyter en HTML-kod till en fördefinierad id.

15 ETT STEG BORTOM HTML:

Dynamisk HTML

Med dynamisk HTML tar hemsidorna definitivt steget från att vara digitala versioner av tidningar till något som närmast liknar interaktiva datorprogram. Dynamisk HTML innehåller inga nya HTML-koder. Istället är det en metod att styra hemsidans delar med hjälp av ett scriptspråk. I det här kapitlet ska vi titta på hur det vanligaste av scriptspråken, JavaScript, fungerar.

Våren 1998 lanserade World Wide Web Consortium (W3C) en ny specifikation för HTML-språket: HTML 4.0. Denna innehöll, förutom alla de vanliga HTML-koder som du lär dig i den här boken, ett antal utökade funktioner som kommit att få samlingsnamnet *dynamisk HTML*, eller *DHTML*.

DHTML bygger på att man med ett scriptspråk, vanligtvis JavaScript, kan styra hur innehållet på hemsidan ser ut och beter sig. Detta är i sig inget nytt, men i och med DHTML har antalet delar av hemsidan som går att styra utökats mycket kraftigt. Det går att med dynamisk HTML i princip manipulera varje aspekt av hemsidan.

Även style sheets, som vi gick igenom i kapitel 14, brukar räknas till dynamisk HTML. Därmed kan DHTML sägas utgöra summan av följande delar:

Vanliga HTML-koder som kan styras med hjälp av ett scriptspråk.
Style sheets, som kan styras på samma sätt.
Scriptspråket som står för styrningen, till exempel JavaScript.

För att förstå hur DHTML fungerar är det nödvändigt att sätta sig in i något av de scriptspråk som utgör teknikens motor. Det finns flera sådana språk att välja mellan t.ex. VBScript och JavaScript. VBScript uppfanns av Microsoft och fungerar bara på företagets egen webbläsare Internet

Tips! När du läst det här kapitlet finns det massor av ytterligare information på Internet. På bokens hemsida hittar du en samling adresser till särskilt viktiga informationssidor.

Explorer. JavaScript utvecklades av Netscape men fungerar även med Internet Explorer och är därför det överlägset mest använda.

I det här kapitlet kommer vi därför helt att koncentrera oss på JavaScript. Språket är dock alldeles för stort för att kunna beskrivas i sin helhet i ett enda kapitel. Det skulle krävas en egen bok. Vad du hittar på följande sidor är istället en grundlig introduktion till språkets centrala delar. Med denna kunskap i bagaget kommer du lätt att kunna dra nytta av de stora mängder information om JavaScript som finns på World Wide Web. Inte minst kommer du att behärska tillräckligt mycket av språket för att kunna anpassa färdiga JavaScript till din egen hemsida. Det är också en bra metod för att bli bättre på JavaScript.

I kapitlet går vi igenom följande:

- Hur man får in JavaScript i HTML-dokumentet.
- JavaScript-språkets syntax.
- Hur JavaScripts objektmodell ser ut och används.
- Hur event handlers används.
- Skillnaden mellan properties och methods.

JavaScript

JavaScript fanns långt innan begreppet dynamisk HTML myntades. Det stöds av Netscape Navigator version 2.0 och senare, samt Internet Explorer 3.0 och senare. Också flera andra modeller av webbläsare, till exempel den utmärkta norska uppstickaren Opera, stöder i någon utsträckning JavaScript.

Tyvär fungerar JavaScript inte likadant i alla webbläsare. De mer avancerade JavaScript-möjligheter som brukar kallas dynamisk HTML fungerar när detta skrivs allra bäst i Internet Explorer 4.0, men även Netscape Navigator 4.0 har ett väl utbyggt JavaScript-stöd. Det mesta som vi går igenom i det här kapitlet fungerar dock även med version 3.0 av båda läsarna.

Viktigt. Förväxla inte JavaScript med programmeringsspråket Java.

JavaScript är inte Java!

Förväxla inte JavaScript med Java, som trots namnet är något helt annat. Java är ett programspråk som Sun Microsystems utvecklat särskilt för användning på Internet. Språket har blivit mycket omtalat därför att det är plattformsoberoende. Det innebär att ett Java-program inte måste kodas i särskilda versioner för varje datortyp, utan fungerar lika bra oavsett om datorn heter Macintosh, PC eller något annat. Java-program ingår inte i själva hemsidan, utan laddas liksom bilder ned som separata filer. Precis som bilder visas programmet sedan i en avgränsad del av webbläsarens fönster. En möjlig framtida utveckling är att vi slipper köpa separata program för till exempel ordbehandling, kalkyler och grafikhantering. Istället kopplar vi upp oss mot olika adresser på WWW och laddar ned ett Java-program som förvandlar webbläsaren till ett fullfjädrat ordbehandlingsprogram, kalkylprogram eller grafikprogram.

Att få in JavaScript i hemsidan

Det finns tre olika sätt att föra in JavaScript-koder i ett HTML-dokument.

- Som attribut inuti de vanliga HTML-koderna
- I separata sektioner, mellan <script> och </script>
- Som *funktioner* i dokumentets <head>-del

JavaScript inuti vanliga HTML-koder

Exempel 15.1 a inehåller ett formulär med bara ett enda objekt: en submitknapp (kapitel 11 beskriver hur formulär fungerar). Vi har försett en av HTML-koderna med en JavaScript-kod.

Exempel 15.1 a
<input /> innehåller både vanliga attribut och JavaScript.

```
<form>
<input type=”submit” value="Klicka mig!" onClick="alert('Aha, ett klick!')" />
</form>
```

Figur 15.1 b
När användaren klickar på knappen visas en *alert,* det vill säga en liten informationsruta.

Titta på HTML-koden <input /> i exemplet. Efter de vanliga attributen type=”submit” och value=”Klicka mig!” kommer något som inte alls liknar HTML: **onClick="alert('Aha, ett klick!')"**. Som du säkert redan gissat är detta JavaScript. Vi återkommer till hur denna kod fungerar, men om du provar kommer du att märka att ett klick på knappen resulterar i en så kallad *alert,* en liten informationsruta (se figur 15.1 b).

JavaScript mellan <script> och </script>

Exempel 15.2 visar den andra metoden för att lägga in JavaScript i ett HTML-dokument: att lägga koden mellan <script> och </script>.

Exempel 15.2
JavaScript kan läggas mellan koderna <script> och </script>.

```
<script language=”javascript”>
alert("Välkommen hit!")
</script>
```

Resultatet av kodningen i exempel 15.2 blir att en meddelanderuta med texten ”Välkommen hit!” visas varje gång sidan laddas in i webbläsaren.

Problem blir det dock om någon som använder en gammal webbläsare, till exempel Internet Explorer 2.0, kommer till din sida. Den läsaren förstår inte <script> och </script> och ignorerar därför dessa koder. Följden blir att webbläsaren uppfattar JavaScript-koden på rad 2 i exemplet, ”alert(”Välkommen hit!”)”, som vanlig text som ska visas på sidan.

För att undvika detta måste JavaScript-koderna alltid omges av kommentarkoder. Exempel 15.3 visar hur det ska se ut:

Exempel 15.3
När JavaScript-koden läggs mellan <script> och </script> måste alltid själva koden omges av kommentarkoder.

```
<body>
<script language=”javascript”>
<!- -
alert("Välkommen hit!")
//-->
</script>
</body>
```

Lägg märke till att den inledande kommentarkoden ser ut precis som vanligt (<!--), men att den avslutande ska föregås av två snedstreck (//).

JavaScript som funktion

Det tredje sättet att få in JavaScript i dokumentet är att lägga det som en så kallad *funktion.* Även här används <script> och </script>, men hela koden måste nu läggas i dokumentets <head>-del. Exempel 15.4 visar hur det går till att skapa en JavaScript-funktion.

Exempel 15.4
JavaScript kan i form av *funktioner* läggas i dokumentets <head>-del. Lägg märke till att kommentarkoderna måste vara med precis som i exempel 15.3.

```
<html>
<head>
<script language=”javascript”>
<!- -
function visaTextruta() {
alert("Välkommen hit...");
alert("...du okände besökare");
}
//- ->
</script></head>
```

En funktion är ett slags ”paket” där en eller flera JavaScript-funktioner ges ett gemensamt namn. Funktionen får namnet ”visaTextruta()”. Parenteserna måste alltid vara med. Funktionen får till uppgift att visa två olika meddelanderutor efter varandra. Lägg märke till att listan över kommandon ska omges av vågparenteser ({ och }) och skiljas åt med semikolon (;).

Det vi gjort i exempel 15.4 är att *definiera* en funktion vid namn visaTextruta. För att den ska åstadkomma någonting måste den nu *anropas.* Hur detta går till ser du i exempel 15.5.

Exempel 15.5
När en funktion väl är definierad kan den när som helst startas från det övriga HTML-dokumentet. Detta sker genom att man helt enkelt anger funktionens namn.

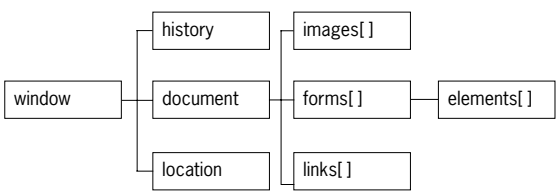
```
<body>
<script language=”javascript”>
<!-
visaTextruta()
//->
</script>
```

Ovanstående kod kommer att starta funktionen visaTextruta, varvid de två meddelanderutorna visas. Det går bra att anropa samma funktion flera gånger. Det kan därför vara tidsbesparande att använda funktioner så mycket som möjligt.

Objektmodellen

Huvudpoängen med JavaScript och dynamisk HTML är att man kan förändra egenskaper och beteenden hos olika bilder, formulärobjekt, textavsnitt och mycket annat på hemsidan. Därför är det viktigt att exakt kunna tala om *vilken* bild eller *vilket* textavsnitt som ska förses med nya egenskaper. För detta ändamål finns något som kallas en *objektmodell.* Med hjälp av denna modell får alla *objekt* på hemsidan automatiskt ett unikt namn.

Figur 15.6 visar några av denna modells viktigaste beståndsdelar. Det finns många fler delar, men av utrymmesskäl har vi bara tagit med de viktigaste.



Figur 15.6
En principskiss över hur objekt-modellen för JavaScript är organiserad. Översikten är av utrymmesskäl långt ifrån komplett. På bokens hemsida hittar du länkar till JavaScript-referenser där hela objektmodellen finns beskriven.

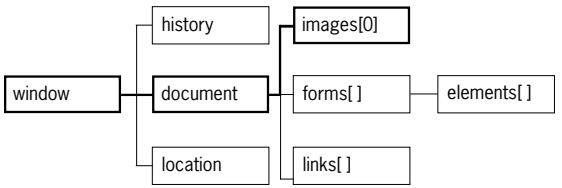
Objektmodellen är *hierarkiskt* uppbyggd. Det innebär att alla objekt på en hemsida har ett namn som speglar dess läge i en hierarki.

Den här koden innehåller till exempel två bilder:

Exempel 15.7
En sida med två bilder.

```
<img src=”bild1.gif” /><br />
<img src=”bild2.gif” />
```

För att ta reda på vad den första bilden heter på JavaScript-språket behöver vi bara följa linjerna i objektmodellen från ”window” som alltid är överst i hierarkin, till ”images[ ]” (*bilder* på svenska):



Figur 15.8
Genom att följa linjerna i objektmodellen ser man varje objekts unika namn.

Namnet på första bilden är enligt JavaScript-språkets sätt att se på saken **window.document.images[0]**. Namnet på den andra bilden blir **window.document.images[1]**. Siffran mellan hakparenteserna numrerar alltså bilderna utifrån den ordning de dyker upp i dokumentet. Första bilden får alltid siffran 0.

Eftersom precis allting som finns på hemsidan ligger under ”window” går det bra att ta bort det steget i namnet. Namnen på de två bilderna blir då **document.images[0]** och **document.images[1]**.

För att det ska vara lättare att snabbt hitta rätt objekt på stora hemsidor kan man också välja att tilldela objekten egna namn:

Exempel 15.9
Enskilda objekt kan också tilldelas speciella namn.

```
<img src="bild1.gif" name="startbild" /><br />
<img src="bild2.gif" name="slutbild" />
```

JavaScript-namnen på dessa bilder blir då betydligt lättare att skilja från varandra: **document.startbild** och **document.slutbild**.

Properties – objektens egenskaper

De flesta av objekten i objektmodellen har en eller flera så kallade ”properties” (*egenskaper* på svenska) knutna till sig. En egenskap som hör till bildobjekten i exempel 15.7 är ”src” och är adressen till själva bilden. Genom att lägga ”src” till bildens JavaScript-namn görs egenskapen manipulerbar:

```
document.images[1].src
```

Om vi någon gång vill ändra adress på bilden räcker det att skriva:

```
document.images[1].src="bilder/nybild.gif"
```

På det här sättet kan bilder bytas ut i samband med att till exempel muspekaren dras över dem, eller när användaren klickar på dem.

Det finns många properties till hemsidans olika objekt, och alla kan ändras med JavaScript. Här är några exempel:

Objekt	Properties
window	.name (fönstrets namn)
	.status (texten i fönstrets statusrad)
window.document	.title (dokumentets titel)
	.bgColor (bakgrundsfärg)
window.location	.href (adressen till hemsidan som visas)

Methods – gör saker med objekten

Förutom properties kan de flesta objekt förses med så kallade *methods* (*metoder* på svenska). En method är som ett slags minimalt program som kan utföra vissa uppgifter. En method som fungerar tillsammans med window-objektet är till exempel **.open()**. Den gör det möjligt för webbläsaren att automatiskt öppna ett nytt fönster och där visa en hemsida.

En viktig method som fungerar tillsammans med dokument-objektet är **.write()**. Med hjälp av det kan webbläsaren fås att skriva saker till sitt eget fönster. Exempel 15.10 visar hur dokument-objektet förses med metoden .write() och skriver en text.

Exempel 15.10
Metoden **.write()** används när man vill att webbläsaren ska skriva till exempel HTML-kod direkt till sitt eget fönster.

```
<script language="javascript">
<!-
document.write("<h1>Skrivet med JavaScript!</h1>")
//->
</script>
```

Resultatet blir att allt som står mellan citationstecknen skrivs ut på sidan. Det går utmärkt att inkludera HTML-kod, och metoden **.write()** används därför ofta till att generera nya hemsidor utifrån besökarens önskemål.

Här är några andra methods:

Objekt	Methods
window	.open(”*url*”) (öppnar ett nytt fönster)
	.close() (stänger ett fönster)
window.location	.replace(”*url*”) (hoppar till en ny sida)
	.reload() (laddar om sidan)
window.history	go(*värde*) (stegar i fönstrets historia)
	back() (går ett steg bakåt)
	forward() (går ett steg framåt)

Att hantera händelser med *event handlers*

Nu har vi bara ett viktigt begrepp kvar att gå igenom innan vi kan börja testa JavaScript i praktiken. Begreppet är *event handlers* (ungefär *händelse-hanterare* på svenska).

Ett event (*händelse* på svenska) är något som händer när en besökare tar del av en hemsida. Det kan vara ett musklick, en tangenttryckning, en musrörelse, en ifyllning av ett formulär och mycket annat. En *event handler* är en mekanism i JavaScript-språket som reagerar på events. När en händelse inträffar är det alltså en händelse-hanterare som reagerar.

Det finns många olika event handlers. Deras namn börjar alltid med ordet ”on” som följs av namnet på det event som event handlern har till uppgift att reagera på. Här följer några andra event handlers, och de event som de är till för att reagera på.

Event handler	**Event som får handlern att reagera**
onClick	När ett element klickas på med musen.
onChange	När inställningen i ett formulär-objekt, till exempel en rullgardinsmeny, ändras.
onSubmit	När ett formulär skickas med submit-knappen.
onMouseOut	När muspekaren rör sig från ett objekt.
onUnLoad	När webbläsaren lämnar en hemsida.

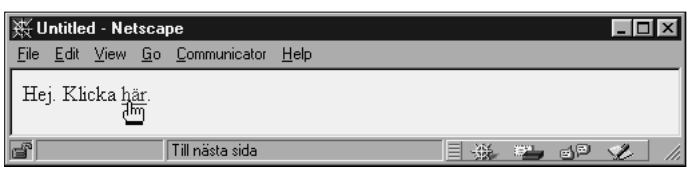
JavaScript-språkets syntax

Nu har vi gått igenom alla de grundbegrepp som man behöver känna till för att använda JavaScript-kod. Men för att kunna använda alltihop måste vi titta på hur de olika delarna ska kombineras, det vill säga JavaScript-språkets syntax.

En vanlig tillämpning av JavaScript är att låta webbläsaren visa en informationstext i fönstrets nederkant då användaren drar muspekaren över en länk. Exempel 15.10 a visar hur koden bakom en sådan effekt och figur 15.10 b visar resultatet.

Exempel 15.10 a
Med onMouseOver kan man bestämma vad som ska visa sig...

```
Hej. Klicka <a href="dok1.html" onMouseOver="window.status=
'Till nästa sida'; return true">här</a>.
```



Figur 15.10 b
...i fönstrets nederkant när muspekaren dras över en länk.

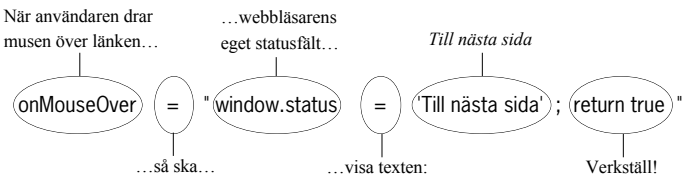
Grunden i koden är en helt vanlig länk. Men till länken har följande JavaScript-rad fogats: **onMouseOver="window.status='Till nästa sida'; return true"**.

Denna rad tolkas ungefär så här av webbläsaren:

När användaren drar musen över länken så ska webbläsarens eget statusfält visa texten Till nästa sida. Verkställ!

För att du lättare ska förstå JavaScript-språkets syntax har vi i figur 15.11 delat upp raden i delar och översatt dessa var för sig.

Figur 15.11
Så här tolkar webbläsaren JavaScript-raden som ingår i exempel 15.10 a. Den fullständiga meningen lyder: "När användaren drar musen över länken så ska webbläsarens eget statusfält visa texten: *Till nästa sida*. Verkställ!"



Vi har redan gått igenom de flesta av kodens beståndsdelar:

onMouseOver är en *event handler* som kontrollerar om användaren placerar muspekaren över länken.

window.status är en kombination av två saker. Dels **window** som är ett objekt, dels **status** som är ett av objektets *properties,* nämligen den text som visas i webbläsarens statusfält. Vanligen visas länkadressen.

'Till nästa sida' är den text som ska visas i statusfältet då muspekaren befinner sig över länken.

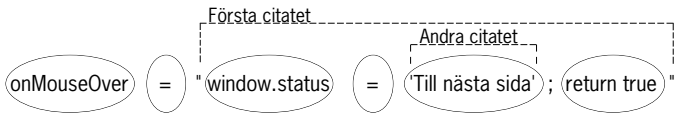
return true betyder ungefär ”Verkställ!”, men används bara undantagsvis i JavaScript-programmering.

De konstiga citationstecknen

I JavaScript lägger man ofta citat inuti varandra. Om man då använder en enda typ av citattecken förstår inte webbläsaren var ett citat börjar och ett annat slutar. Man måste därför alternera mellan dubbla och enkla tecken.

I figur 15.12 markeras det första citatet med dubbla citattecken. Därför måste det andra citatet, som ligger inuti det första, markeras med enkla tecken. Det går bra att lägga ännu fler citat i varandra, bara man hela tiden alternerar mellan dubbla och enkla tecken.

Figur 15.12
När citat ligger i varandra måste citattecknen alternera mellan dubbla (") och enkla (').



Genom att lägga JavaScript-raden i exempel 15.10 a till alla dina länkar kan du visa dina besökare en individuell informationstext för varje länk. I vissa webbläsare försvinner dock inte texten automatiskt när muspekaren inte längre befinner sig över länken. Detta problem löser man genom att lägga till event handlern **onMouseOut** till koden:

Exempel 15.13
Event handlern onMouseOut ser till att texten i statusfältet försvinner när muspekaren lämnar länken.

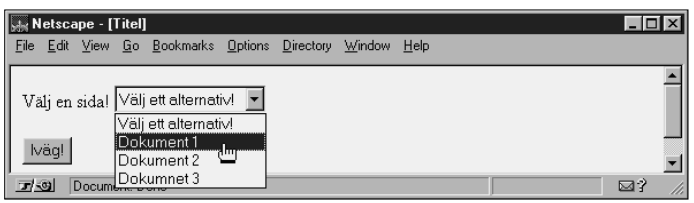
```
Hej. Klicka <a href="dok1.html" onMouseOver="self.status='Till
nästa sida'; return true" onMouseOut="self.status='  '; return
true">här</a>.
```

Lägg märke till att det som följer self.status= bara är några mellanslag inom citationstecken. När webbläsaren visar dessa mellanslag blir effekten att texten i statusraden raderas.

En meny i JavaScript

Nästa exempel är mycket användbart för dig som har många länkar på din hemsida. Med hjälp av ett JavaScript samlar du enkelt så många länkar du vill i en praktisk rullgardinsmeny. När besökaren gjort sitt val trycker hon på en knapp, och webbläsaren hoppar till den valda länken. Figur 15.14 a visar hur menyn ser ut. Exempel 15.14 b visar koden, och nedan går vi igenom de viktigaste raderna.

Figur 15.14 a
En praktisk meny som låter användaren välja vilken sida hon vill hoppa till.



Exempel 15.14 b
Koden till länkmenyn. De tre raderna som börjar på "if" styr vart varje menyalternativ ska leda. I exemplet leder första alternativet (som har nummer 0) ingenstans eftersom det bara innehåller texten "Välj ett alternativ!" (se figur 15.14 a).

```
1  <html>
2  <head><title>Titel</title>
3  <script language="javascript">
4  <!- -
5  function javameny() {
6  if (document.forms[0].elements[0].options[1].selected == true)
7  window.location='dok1.html';
8  if (document.forms[0].elements[0].options[2].selected == true)
9  window.location='dok2.html';
10 if (document.forms[0].elements[0].options[3].selected == true)
11 window.location='dok3.html';
12 }
```



```
13 //- ->
14 </script></head>
15 <body>
16 <form name="formular">Välj en sida!
17 <select>
18 <option>Välj ett alternativ!</option> <option>Dokument
19 1</option> <option>Dokument 2</option> <option>Dokument
20 3</option>
21 </select><p>
22 <input type="button" value="Iväg!" onClick="javameny()">
23 </form>
24 </body>
25 </html>
```

De viktigaste raderna i exempel 15.14 a

5. Här skapar vi en funktion som har två uppgifter: dels att ta reda på vilket alternativ användaren valt i menyn, dels att få webbläsaren att hoppa till den valda sidan. "function javameny()" skapar funktionen och ger den namnet "javameny". Vågparentesen ({) betyder att det som följer är en lista över uppgifter som ska ingå i funktionen.

6–7. Här kommer en så kallad *if-sats*. Det är en instruktion till webbläsaren att utföra en viss uppgift, men bara under vissa villkor. If-satser är uppbyggda så här:

om (*villkor*) så gör det här.

Rad 6 och 7 betyder således

om (*just detta menyalternativ valts*) så hoppa till dok1.html

Varje alternativ i menyn har en egenskap som heter "selected". Denna kan ha två värden: *true* (sann) eller *false* (falsk). Det menyalternativ som är valt får ett selected-värde som är *true*. På rad 6 ställs villkoret att selected ska vara lika med ("==" i javaScript-språket) *true*. Om så är fallet hoppar webbläsaren till den *location* som står angiven på rad 7. I annat fall hoppar webbläsaren vidare till rad 8 för att se om villkoret i den if-satsen är uppfyllt.

8–15. Här upprepas samma villkor för de övriga två alternativen i menyn (egentligen innehåller menyn fyra alternativ, men det första, "Välj ett alternativ!", ska inte leda någonstans). Slutligen avslutas innehållet i funktionen med vågparentes (}), <script> och <head> avslutas och <body>-delen av dokumentet inleds.

16–23. Till själva rullgardinsmenyn används ett formulär av select-typ. När användaren gjort sitt val och klickar på knappen startar event handlern *onClick* funktionen javameny. Därefter hoppar webbläsaren till det dokument besökaren valt.

Sugen på mer JavaScript?

Du har nu stiftat bekantskap med grundbegreppen i JavaScript. Ett bra sätt att lära sig mer är att försöka få några av de många färdigskrivna JavaScript-kodningar som finns på World Wide Web att fungera på din egen hemsida. På bokens hemsida (www.etc.se/nyckeln/) hittar du bland annat följande färdiga exempel:

- JavaScript som kollar vilken webbläsare besökaren har. Den som använder en för gammal version får ett varningsmeddelande.
- Bakåt- och framåtknappar i JavaScript. Användbara när du vill att dina besökare enkelt ska kunna stega fram och tillbaka i ett rutsystem.
- JavaScript som byter ut bilder då besökaren drar musen över dem.

På hemsidan hittar du dessutom en samling bra adresser med mer information om konsten att skriva egen JavaScript-kod.

Dreamweaver

Måste man verkligen lära sig JavaScript för att kunna använda alla nya finesser i dynamisk HTML? Nej, lyckligtvis inte.

Dreamweaver heter en HTML-editor som förutom vanlig HTML också klarar dynamisk HTML. Dreamweaver är en så kallat **wysiwyg-editor** (uttalas *wyssiwigg*). Det innebär att det går att i ett särskilt fönster rita upp hur hemsidan ska se ut, och sedan överlåta till programmet att generera den rätta HTML-koden. I praktiken är dock Dreamweaver svårt att använda för den som inte kan HTML. Det finns mycket som kan gå fel, och den som då inte klarar att manuellt rätta till koden kör lätt fast.

Men för den som behärskar den konsten (till exempel efter att ha läst den här boken) är programmet ett utmärkt hjälpmedel.

När detta skrivs är Dreamweaver fortfarande ett alldeles för dyrt program för privatpersoner. Under tiden finns det en gratis testversion som kan användas i 30 dagar att ladda ned från www.macromedia.com.

? Wysiwyg. Förkortning av "What you see is what you get", vilket ungefär betyder "Vad du ser är vad du får". Ett wysiwyg-program visar en exakt bild av det färdiga resultatet på skärmen. Desktop publishing-program är exempel på wysiwyg-program, eftersom de ju på skärmen visar hur till exempel tidningen som layoutas kommer att se ut i tryck.

KODER FÖR SÄRSKILDA LÄSARE

16 Internet Explorer

Version 3.0 och 4.0 av Internet Explorer stöder en del koder som inte Netscape Navigator förstår. Om du vet att dina besökare använder Internet Explorer kan du därför experimentera med finesser som rörlig text och bakgrundsljud. I det här kapitlet går vi igenom alla koder som är unika för Internet Explorer.

Netscape och Microsoft har länge tävlat om användarna genom att hitta på egna HTML-koder. Tanken är att du som gör hemsidor ska bli lockad av de nya finesserna och göra sidor som bara kan ses med ena företagets webbläsare. Ofta snappar företagen efterhand upp varandras innovationer, men det är oundvikligt att det vid varje givet tillfälle finns ett antal koder som bara en enda webbläsare förstår.

I det här kapitlet ska vi titta på koder som bara fungerar om de läses med Internet Explorer 3.0 och 4.0. Netscape Navigator kommer helt att ignorera koderna. Oftast kommer innehållet att bli begripligt ändå, eftersom Microsoft försökt göra sina nya koder någorlunda kompatibla även med äldre webbläsare. Besökare som använder Netscape Navigator kommer dock att se dina sidor på ett helt annat sätt än det var tänkt.

Om du vill använda specialkoderna i det här kapitlet är det därför viktigt att du testkör sidorna även i Netscape Navigator. Ett alternativ är att tydligt tala om att dina sidor bara ska ses med Internet Explorer 3.0 och 4.0. Det kommer dock att minska antalet potentiella besökare kraftigt.

Att Netscape Navigator för närvarande inte stöder de här koderna betyder inte att det alltid kommer att vara så. Organisationen W3C har inkluderat de flesta av Internet Explorers tillägg i sin senaste standard för HTML-språket, HTML 4.0. Eftersom Netscape lovat att respektera denna standard i framtiden, lär koderna ingå i framtida versioner av Navigator.

Viktigt.
Webbläsare som inte förstår <marquee>-koden kommer att visa själva texten i stillastående form. Om texten är för lång kommer dock inte all text att få plats.
Du kan alltså använda <marquee> även om du inte är säker på att alla besökare har Internet Explorer, under förutsättning att texten som rullar är tillräckligt kort. En bra regel är att alltid testa hur sidan ser ut sedd med Netscape Navigator.

Rullande text

Förr var det nödvändigt att använda krångliga CGI-program för att få text att röra sig på hemsidan. Inte nu längre. Placera bara en text mellan koderna <marquee> och </marquee> , och den rör sig från höger till vänster i en evig loop. Effekten påminner om en ljusskylt.

Det finns en lång rad frivilliga attribut som styr hur den rullande texten beter sig och hur rutan texten rullar i ska se ut och justeras. Här är allihop:

bgcolor=*"#färgkod"* ger den rullande texten en egen bakgrundsfärg, vilket förstärker effekten av ljusskylt.

Height=*"värde"* och width=*"värde"* talar om hur stor rutan ska vara.

align=*"top", "middle"* och "*bottom"* justerar rutan i förhållande till textraden. Attributet fungerar exakt som align i samband med bilder (se figur 6.6 b i kapitel 6).

Vspace=*"värde"* och hspace=*"värde"* talar precis som för bilder om hur mycket luft texten ska omges av vertikalt och horisontellt.

Med direction=*"left"* eller *"right"* bestämmer du åt vilket håll texten ska röra sig, och med behavior=*"scroll", "slide"* eller *"alternate"* på vilket sätt den ska röra sig. *Scroll* innebär att texten kommer in från ena sidan och helt försvinner på andra sidan innan rörelsen upprepas. *Slide* gör så att texten kommer in från sidan och stannar när den rör vid marginalen på motsatta sidan. *Alternate* innebär att texten studsar fram och tillbaka.

Med loop=*"värde"* bestämmer du hur många loopar texten ska göra. Grundinställningen är ett oändligt antal.

Scrollamount=*"värde"* bestämmer slutligen hur många pixlar texten ska röra sig åt gången, och scrolldelay=*"värde"* anger hur många millisekunder det ska gå mellan varje rörelse.

Exempel 16.1 a visar hur <marquee> används.

Exempel 16.1 a
En svart rullande text på grå botten. Lägg märke till att <font face="impact"> ska placeras utanför <marquee>-koderna för att ändra typsnitt på texten.

```
<font face="impact" size="7">
<marquee bgcolor="#999999" behavior="scroll" direction="left"
height="60" width="500" hspace="10" vspace="3" scrolla-
mount="3" scrolldelay="5" vspace="5"><font size="7">Nyckeln
till din egen hemsida på Internet</marquee>
```

Figur 16.1 b
Resultatet. Texten rör sig från höger till vänster i en grå ruta.

Flytande rutor

I kapitlet om rutor (kapitel 10) lärde du dig att ett rutsystem alltid måste definieras i ett särskilt <frameset>-dokument. Microsoft har hittat på ett sätt att slippa den processen och istället inkludera en ruta direkt i HTML-dokumentet ungefär som om den var en bild.

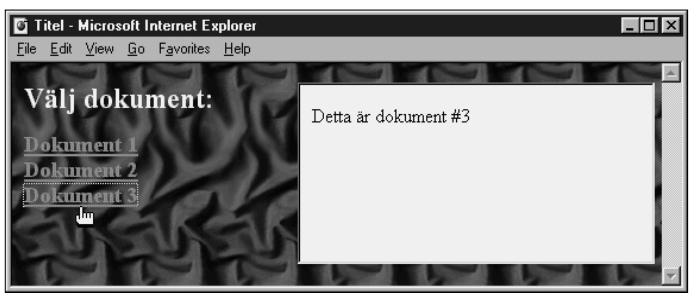
Koden som gör detta heter <iframe>...</iframe>. Rutan fungerar exakt som en <frame> och använder samma attribut (se kapitel 10 för mer detaljerad information om attributen):

frameborder=*"0"* eller *"1"*	Slår på och av ramen.
marginheight=*"värde"*	Anger vertikal marginal.
marginwidth=*"värde"*	Anger horisontell marginal.
name=*"namn"*	Ger rutan ett namn så att den kan ta emot länkar.
scrolling=*"yes"* eller *"no"*	Slår på och av rullisterna.

<iframe> har dessutom attributet align=*"top", "middle", "bottom", "left"* och *"right"*. De tre första värdena bestämmer justering i förhållande till en textrad, medan *left* och *right* gör så att text flyter runt rutan. <iframe> beter sig alltså exakt som en bild, och mer information om alignvärdena hittar du i kapitel 6. I exempel 16.2 a skapas en ruta som flyter till höger. För att den ska synas ordentligt har sidan fått en bakgrundsbild.

Exempel 16.2 a
En flytande ruta fungerar precis som en <frame />, men utan <frameset>. Den här rutan heter "dokruta", de klickbara länkarna leder till den med attributet taget="dokruta".

```
<iframe name="dokruta" align="right" src="dok1.html">
</iframe>
<h2>Välj dokument:</h2>
<font size="4"><b>
<a href="dok1.html" target="dokruta">Dokument 1</a><br />
<a href="dok2.html target="dokruta">Dokument 2</a><br />
<a href="dok3.html target="dokruta">Dokument 3</a>
</b></font>
```



Figur 16.2 b
Raultatet av kodningen i exempel 16.2 a. För att rutan ska synas har sidan fått en bakgrundsbild. Det går bra att ha flera flytande rutor i ett dokument.

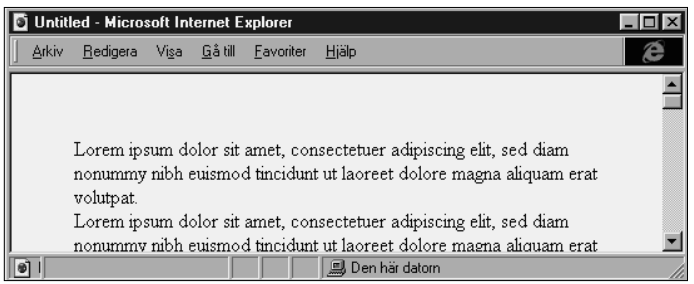
Skapa marginaler

Microsoft har lagt till möjligheten att skapa en marginal till vänster och överst på sidan. Detta görs med hjälp av två nya attribut till <body>-koden. Topmargin=*"värde"* skapar en övre marginal av valfri pixelhöjd, leftmargin=*"värde"* skapar en vänstermarginal.

Exempel 16.3 a och figur 16.3 b visar hur de nya attributen används.

Exempel 16.3 a
Med attributen leftmagin och topmargin skapas här 50 pixlar bred marginal överst och till vänster på sidan.

```
<body leftmargin="50" topmargin="50">
Lorem ipsum dolor (osv)
</body>
```



Figur 16.3 b
Resultatet av kodningen i exempel 16.3 a.

Fixerad bakgrundsbild

Ett annat nytt attribut till <body> är bgproperties=*"fixed"* . Det gör så att sidans bakgrundsbild ligger stilla när sidans övriga innehåll rullas upp eller ned med rulllisten. Det ger en tredimensionell effekt som inte går att visa med en stillbild. Testa därför följande kod på din hemsida.

```
<body background="sökväg" bgproperties="fixed">
```

Bakgrundsljud

Om du har ett ljud som du tycker att alla bör lyssna på då de besöker din hemsida skriver du så här någonstans i dokumentet:

```
<bgsound src="sökväg till ljudfilen" loop="värde" eller "infinite">
```

Loop="värde" anger hur många gånger ljudfilen ska spelas upp. Om du verkligen gillar ljudet gör loop="infinite" att det spelas oavbrutet tills besökaren lämnar sidan. Om loop utelämnas spelas ljudet upp en gång. Ljudfilen måste vara i WAV, AU eller MIDI-format.

Tänk på att dina besökare aldrig bett om att få höra bakgrundsljud, och att de därför kan bli frustrerade av att tvingas vänta på att stora ljudfiler ska laddas ned. Ett bakgrundsljud bör absolut inte vara större än 100 KB.

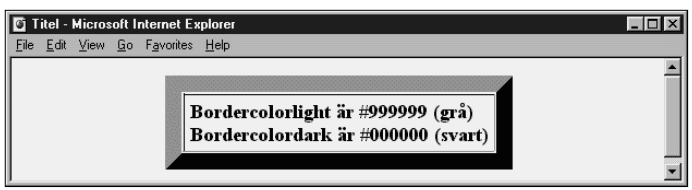
Bestäm färg på tabellramar

Tre nya attribut till <table> styr färgen på tabellens ramar och linjerna mellan cellerna. Enklast att använda är bordercolor=*"#färgkod"*, som helt enkelt ger den valda färgen till alla tabellens ramar och linjer. För att ramarna ska synas måste även border eller border=*"värde"* anges.

I normala fall är en ram uppbyggd av två olika färger för att skapa en tredimensionell effekt. Ett mer avancerat alternativ till bordercolor är att definiera dessa färger separat. Attributen heter bordercolorlight="*#färgkod"* (ljusa färgen) och bordercolordark="*#färgkod"* (mörka färgen). I exempel 16.4 a har grått och svart används. Figur 16.4 b visar resultatet.

Exempel 16.4 a
Med attributen bordercolorlight och bordercolordark styr du vilken ljus och mörk färg som ska ingå i ramen.

```
<table cellpadding="5" border="15" bordercolor-
light="#999999" bordercolordark="#000000">
<tr><td> <b>Bordercolorlight är #999999 (grå)<br />
Bordercolordark är #000000 (svart)</td></tr>
</table>
```



Figur 16.4 b
Resultatet. För att ge en 3D-effekt visas den ljusa färgen överst och till vänster, och den mörka nederst och till höger.

Viktigt.
Microsofts nya tabellorganisation är skapad för att fungerar även i Netscape Navigator och andra äldre webbläsare. Den kommer dock inte att se lika bra ut som i Internet Explorer 3.0 och 4.0.

En ny tabellorganisation

Om du använder tabeller till att presentera information, snarare än till layout, kan Microsofts nya sätt att organisera tabellen vara användbart. Särskilt gäller detta stora tabeller. Tanken är att du genom att dela in tabellen i sektioner som sedan kan tilldelas specifika egenskaper ska slippa definiera hur varje enskild cell ska bete sig.

Koderna <thead>, <tbody> och <tfoot> används för att skapa horisontella sektioner, och för vertikala sektioner används <colgroup>. <col> styr egenskaperna för enskilda kolumner.

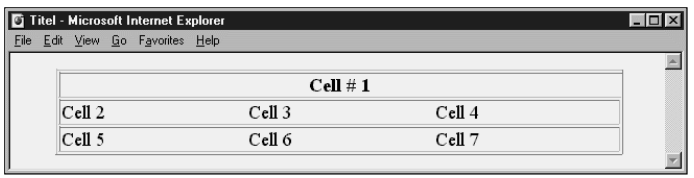
Horisontella grupper

Dessa koder saknar avslutningskoder och attribut, och sprängs in i tabellen för att ge den ett tabellhuvud (<thead>), en kropp (<tbody>) och en tabellfot (<tfoot>). Med hjälp av ett nytt attribut till <table>, rules=*"groups"* (mer om detta längre fram), får de olika avdelningarna linjer mellan sig.

Tabellen i exempel 16.5 a har delats upp i tre sektioner med <thead>, <tbody> och <tfoot>. Rules=*"groups"* i <table> skapar linjer mellan avdelningarna. Figur 16.5 b visar resultatet. Lägg märke till att de enskilda cellerna nu inte har några egna ramar.

Exempel 16.5 a
Tabellen får tre avdelningar med <thead>, <tbody> och <tfoot>. Attributet rules=*"groups"* i <table> gör att de tre avdelningarna skiljs åt av linjer. Lägg märke till att attributet border måste användas för att linjerna ska synas.

```
<table rules="groups" border width="90%">
<thead>
<tr>
<th colspan="3">cell # 1</th></tr>
<tbody>
<tr><td>cell 2</td><td>cell 3</td><td>cell 4 </td></tr>
<tfoot>
<tr><td>cell 5</td><td>cell 6</td><td>cell 7 </td></tr>
</table>
```



Figur 16.5 b
Resultatet av kodningen i exempel 16.5 a. Lägg märke till att de enskilda cellernas ramar inte syns.

Vertikala grupper

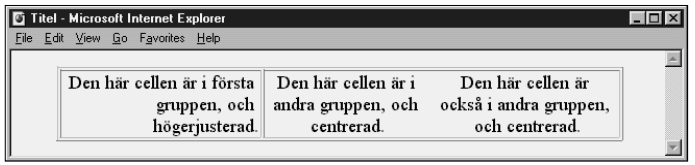
Med <colgroup> delar du in tabellens kolumner i grupper, som sedan precis som i förra exemplet skils åt med linjer genom attributet rules=*"groups"*, som anges i <table>. <colgroup> saknar avslutningskod och placeras före en ny rad (<tr>). Den har två attribut.

Span=*"värde"* anger hur många kolumner gruppen ska omfatta, och align=*"left", "center"* eller *"right"* hur innehållet i gruppens samtliga celler ska justeras.

Exempel 16.6 a visar en tabell där två kolumngrupper definieras. Den första innehåller bara en kolumn (genom att span=*"värde"* utelämnats), medan den andra tilldelas två kolumner med attributet span=*"2"*. Innehållet i de två kolumngrupperna får olika justering med align=*"right"* respektive *"center"*. Figur 16.6 b visar resultatet.

Exempel 16.6 a
Med hjälp av <colgroup> skapas kolumngrupper. Resultatet syns i figur 16.6 b på nästa sida.

```
<table rules="groups" border width="90%">
<colgroup align="right">
<colgroup span="2" align="center">
<tr>
<td>Den här cellen är i första gruppen, och högerjusterad.</td>
<td>Den här cellen är i andra gruppen, och centrerad.</td>
<td>Den här cellen är också i andra gruppen, och
centrerad.</td></tr></table>
```

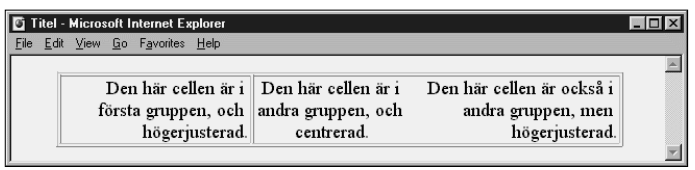
Figur 16.6 b
Resultatet av kodningen i exempel 16.6 a. Lägg märke till att rules=*"groups"* nu skapar linjer mellan kolumngrupperna. Jämför gärna med figur 16.5 b.

Styr enskilda kolumner

Om du inte vill ha samma justering i alla kolumner inom en grupp kan du med <col> ange specifika egenskaper för varje kolumn. <col> använder samma attribut som <colgroup> det vill säga align=*"left", "center"* eller *"right"* och span=*"värde"*. Exempel 16.7 a visar hur <col> kompletterar <colgroup>. Jämför gärna med koden i exempel 16.6 a, och lägg särskilt märke till att attributet span=*"värde"* inte längre behöver användas då den andra kolumngruppen skapas med <colgroup>. Att gruppen innehåller två kolumner framgår istället av att två <col>-koder följer direkt efter <colgroup>.

Exempel 16.7 a
Med <col> kan cellerna i en enskild kolumn få egen justering.

```
<table rules="groups" border width="90%">
<colgroup align="right">
<colgroup>
  <col align="center">
  <col align="right">
<tr>
<td>Den här cellen är i första gruppen, och högerjusterad.</td>
<td>Den här cellen är i andra gruppen, och centrerad.</td>
<td>Den här cellen är också i andra gruppen, men högerjusterad.</td>
</tr>
</table>
```



Figur 16.7 b
Resultatet av kodningen i exempel 16.7 a.

Rules-attributet

I de två sista exemplen har vi använt det nya attributet rules för att skapa linjer mellan horisontella sektioner och kolumngrupper. Men rules har fler attribut, och nu ska vi gå igenom allihop. Rules kan bara användas som attribut till <table>. Lägg märke till att rules bara påverkar linjerna inuti tabellen, inte dess yttre ram. För det används attributet frame som vi går igenom nedan.

rules=*"none"*	Inga linjer inne i tabellen visas. Tabellens yttre ram påverkas dock inte.
rules=*"groups"*	Linjer visas mellan grupper definierade med <thead>, <tbody>, <tfoot> och <colgroup>.
rules=*"rows"*	Horisontella linjer visas mellan alla rader.
rules=*"cols"*	Vertikala linjer visas mellan alla kolumner.
rules=*"all"*	Linjer visas mellan alla rader och kolumner.

FRAME-attributet

Medan rules bara bestämmer vilka linjer som ska visas inuti tabellen, styr det nya attributet frame tabellens yttre ram. Frame kan bara användas som attribut till <table> och har följande värden:

frame=*"void"*	Ingen yttre ram visas.
frame=*"above"*	Ramen visas bara på tabellens överkant.
frame=*"below"*	Ramen visas bara på tabellens nederkant.
frame=*"hsides"*	Ramen visas på ramens över- och nederkant.
frame=*"lhs"*	Ramen visas bara på tabellens vänstersida.
frame=*"rhs"*	Ramen visas bara på tabellens högersida.
frame=*"vsides"*	Ramen visas på ramens vänster- och högersida.
frame=*"box"*	Ramen visas runt hela tabellen.
frame=*"border"*	Samma som frame=*"box"*.

Det är fritt fram att kombinera frame- och rules-attributen i <table> med <thead>, <tbody>, <tfoot> och <colgroup>. På så sätt kan du åstadkomma tabeller som till exempel saknar yttre ram, men som visar linjer mellan horisontella och/eller vertikala sektioner. Tänk bara på att attributet border (eller border=*"värde"* om du vill ha en tjockare ram) alltid måste finnas med i <table> för att ramar och linjer ska bli synliga.

Viktigt. Samtliga koder i det här kapitlet är speciellt avsedda för att ses med Microsoft Internet Explorer 3.0 och 4.0. Oftast kommer koderna att fungera hjälpligt även i Netscape Navigator och andra läsare, men testa för säkerhets skull alltid din sida med dessa läsare om du inte är säker på att alla dina besökare använder Internet Explorer.

Koderna i sammanfattning

<marquee>…</marquee>

Skapar en ruta med rullande text.

Attribut till <marquee>

align=*"left"*, *"center"* eller *"right"*
Bestämmer rutans horisontella justering.

behavior="*scroll"*, *"slide"* eller *"alternate"*
Bestämmer textens sätt att röra sig.

bgcolor=*"#färgkod"*
Anger textens bakgrundsfärg.

direction=*"left"* eller *"right"*
Bestämmer textens rörelseriktning.

height=*"värde"* eller *"värde%"*
Anger rutans höjd i pixlar eller procent av fönstrets höjd.

width=*"värde"* eller *"värde%"*
Anger rutans bredd i pixlar eller procent av fönstrets bredd.

hspace=*"värde"*
Anger horisontell marginal utanför rutan.

vspace=*"värde"*
Anger vertikal marginal utanför rutan.

loop=*"värde"*
Anger antalet loopar texten ska göra.

scrollamount=*"värde"*
Anger hur många pixlar texten ska röra sig per steg.

scrolldelay=*"värde"*
Anger antal millisekunder mellan varje steg.

<iframe>…</iframe>

Skapar en flytande ruta.

Attribut till <iframe>

align=*"top"*, *"center"*, *"middle"*, *"left"* eller *"right"*
Anger rutans justering.

frameborder=*"1"* eller *"0"*
Slår på eller av rutans ram.

height=*"värde"*
Anger rutans höjd i pixlar.

width=*"värde"*
Anger rutans bredd i pixlar.

marginheight=*"värde"*
Anger rutans övre och undre marginal i pixlar.

marginwidth=*"värde"*
Anger rutans vänster- och högermarginal i pixlar.
name=*"namn"*
Ger rutan ett namn så att den kan ta emot länkar.
scrolling=*"yes"* eller *"no"*
Slår på och av rullisten.
src=*"sökväg"*
Anger vilket html-dokument som ska visas i rutan.

NYA FUNKTIONER I TABELLER

<thead>...</thead>
Skapar ett tabellhuvud.
<tbody>
Skapar en tabellkropp.
<tfoot>...</tfoot>
Skapar en tabellfot.
<colgroup>...</colgroup>
Skapar en grupp av kolumner.

attribut till <colgroup>

span=*"värde"*
Anger hur många kolumner gruppen omfattar.
align=*"left"*, *"center"* eller *"right"*
Anger hur cellinnehållet i gruppens kolumner ska justeras.

<col>...</col>
Styr en enskild kolumn i en kolumngrupp. <col> har samma attribut som <colgroup> (se ovan).
<table>…</table>
Skapar en tabell.

Nya attribut till <table>

rules=*"none"*, *"groups"*, *"rows"*, *"cols"* eller *"all"*
Bestämmer vilka delar av tabellen som ska skiljas åt med linjer.
frame=*"void"*, *"above"*, *"below"*, *"hsides"*, *"vsides"*, *"lhs"*, *"rhs"*, *"box"*, *"border"*
Bestämmer hur tabellens yttre ram ska se ut.

ÖVRIGA NYHETER

Nya attribut till <body>

topmargin=*"värde"* och leftmargin=*"värde"*
Skapar en marginal överst respektive till vänster på sidan. värdet anges i pixlar.
bgproperties=*"fixed"*
Ger stillastående bakgrundsbild.

KODER FÖR SÄRSKILDA LÄSARE

17 Netscape

Netscape och Microsoft tävlar om vem som kan hitta på de bästa koderna till just sin webläsare. Nackdelen är att många koder bara fungerar med en av webläsarna. I det här kapitlet går vi igenom de HTML-koder som Netscape ännu så länge är ensam om.

Netscape har länge konstruerat egna HTML-koder som är så bra att en stor del av världens HTML-kodare valt att göra hemsidor enbart för Netscape Navigator. Nu har Microsoft svarat med att integrera nästan alla Netscapes specialkoder i sin egen webläsare Internet Explorer, och dessutom hittat på egna, unika koder. De ska vi titta på i nästa kapitel.

I det här kapitlet ska vi gå igenom de relativt få av Netscapes specialkoder som Microsoft Internet Explorer ännu inte förstår. Den kraftfullaste låter dig presentera text i spaltform, något som inte tidigare varit möjligt. Ett annat välkommet tillägg är koden <spacer>, med vilken du enkelt skapar tomma utrymmen på din sida.

Även om koderna i det här kapitlet inte fungerar med Internet Explorer 3.0 och 4.0 är det inte omöjligt att Microsoft inkluderar en eller flera av dem i en framtida version av programmet.

Viktigt.
Om du väljer att använda koderna i det här kapitlet bör du tydligt tala om för dina besökare att de måste använda Netscape Navigator 3.0 eller 4.0 för att sidan ska se bra ut. Senaste versionen av Netscape Navigator kan laddas ned på adress http://home.netscape.com/comprod/mirror/.

Visa text i flera spalter

En svaghet hos HTML-språket har varit oförmågan att visa text i mer än en spalt. Det problemet har Netscape nu löst med koden <multicol>...</multicol> (efter engelskans *multicolumns,* flera spalter). Texten mellan dessa båda koder visas i flera spalter, och med det obligatoriska attributet cols=*”värde”* anger du hur många spalter du vill ha. Därutöver finns två frivilliga attribut. Gutter=*”värde”* anger hur mycket utrymme det ska finnas mellan spalterna. Om gutter utelämnas blir

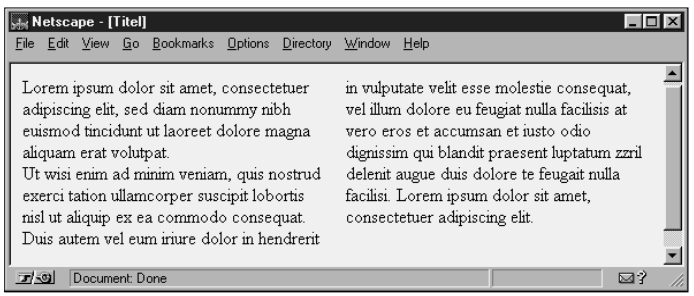
avståndet automatiskt 10 pixlar. width=*"värde"* eller *"värde%"* anger hur bred hela texten ska vara mätt i pixlar eller som procent av webläsarens fönsterbredd. Om ingen bredd anges blir värdet automatiskt 100 %.

Exempel 17.1 a visar hur <multicol> används. Texten visas i två spalter med 15 pixlars mellanrum. Figur 17.1 b visar hur resultatet blir i Netscape Navigator 3.0.

Exempel 17.1 a
Med koden <multicol> kan text visas i flera spalter. Resultatet av kodningen ser du i figur 17.1 b.

```
<multicol cols="2" gutter="15" width="100%">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat.<br />
(...)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
</multicol>
```

Figur 17.1 b
Resultatet av kodningen i exempel 17.1 a. Lägg märke till att Netscape Navigator försöker dela upp texten jämnt mellan de två spalterna.



Netscape Navigator försöker göra alla spalter lika långa. I exemplet ovan är radantalet dock udda, varför en spalt blir längre än den andra.

Det går bra att lägga flera <multicol> i varandra. På så sätt kan du låta delar av en spalt delas upp i ytterligare nya spalter. Du kan på det sättet

En kod som skapar luft

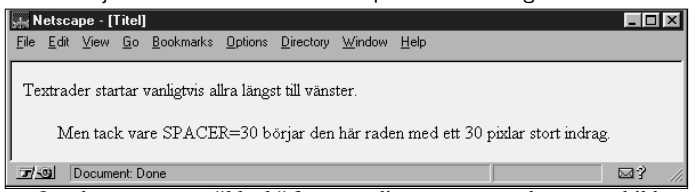
kombinera ett obegränsat antal <multicol>.
Det har tidigare inte funnits något riktigt bra och enkelt sätt att skapa luft mellan de olika objekten på en hemsida. Detta har Netscape ordnat med koden <spacer>. Den saknar avslutningskod och har helt enkelt till uppgift att utan att synas ta upp en viss plats på sidan, horisontellt, vertikalt eller i form av en rektangel. Med attributet type=*"horizontal", "vertical"* eller *"block"* anger du vilken form din <spacer> ska ha. Size=*"värde"* talar om hur mycket plats <spacer> ska ta om type är satt till *"horizontal"* eller *"vertical"*. Exempel 17.2 a visar hur en horisontell <spacer> används, och

Exempel 17.2 a
En 30 pixlar stor horisontell <spacer> i aktion.

Figur 17.2 b
Resultatet. Raden trycks in 30 pixlar. En <spacer> är alltid helt osynlig.

resultatet syns i figur 17.2 b.

```
Textrader startar vanligtvis allra längst till vänster.<p>
<spacer type="horizontial" size="30">Men tack vare SPACER=30
börjar den här raden med ett 30 pixlar stort indrag.
```



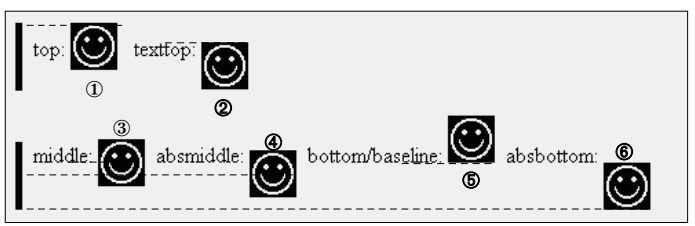
Om du anger type=*"block"* fungerar din <spacer> exakt som en bild. Du måste då ange width=*"bredd"*, height=*"höjd"*. Även align=*"left"* eller *"right"* kan användas (se kapitel 6 för mer information om align).

Nya sätt att justera bilder

I kapitel 6 gick vi igenom hur man med hjälp av attributet align=*"top", "middle"* och *"bottom"* kan påverka en bilds placering i förhållande till en textrad (se figur 6.6 b i kapitel 6). Netscape har kompletterat align-attributet med fyra egna värden: *"texttop", "baseline", "absmiddle"* och *"absbottom"*. Align=*"baseline"* är dock bara ett nytt och mer passande namn på det gamla align=*"bottom"*. För att förstå vad de nya värdena gör ska vi jämföra dem med de gamla. Siffrorna hänvisar till figur 17.3 nedan.

① align=*"top"* linjerar bildens överkant med det högsta objektet på raden, vilket kan vara textens överkant eller en annan bild.
② align=*"texttop"* linjerar alltid bildens överkant med radens högsta text.
③ align=*"middle"* linjerar bildens mittlinje med textens baslinje.
④ align=*"absmiddle"* linjerar bildens mittlinje med mittlinjen hos radens största objekt, vilket även kan vara en bild.
⑤ align=*"bottom/baseline"* linjerar bildens nederkant med textens baslinje.
⑥ align=*"absbottom"* linjerar bildens nederkant med nederkanten på radens lägsta objekt, vilket även kan vara en bild.

Figur 17.3
Netscape har utökat align-attributet med tre nya värden och döpt om *bottom* till *baseline*. Bilden visar hur både de gamla och de nya värdena används.



Viktigt.
<layer>–koderna ger en del nya möjligheter som style sheets ännu inte kan erbjuda. Trots detta är det nästan alltid bäst att använda style sheets för att skapa lager (se kapitel 14). Då fungerar ju sidorna med både Netscape Navigator och Internet Explorer. Om du vet att dina besökare använder Netscape Navigator 4.0 eller senare versioner kan dock <layer>-koderna vara ett alternativ.

Lager med <layer>

I kapitel 14 utnyttjade vi style sheet-tekniken för att lägga hemsidans innehåll i flera lager ovanpå varandra. Netscape har uppfunnit två HTML-koder som åstadkommer samma sak utan att man behöver använda style sheets: <layer>...</layer> och <ilayer>...</ilayer>.
Allt som placeras mellan <layer> och </layer> ingår i ett lager. Attributen left=*"pixelposition"* och top=*"pixelposition"* anger exakt var lagret ska placeras räknat från webläsarens vänstra och övre kant. Med z-index=*"värde"* talar du om var lagret ska ligga i djupled. Ett högt värde placerar lagret högre upp.

Lagrets storlek bestäms med height=*"värde"* och width=*"värde"*. Och precis som i tabeller kan bakgrundsfärg och bakgrundsbild bestämmas med bgcolor=*"färgkod"* respektive background=*"sökväg till bild"*. Om ingen bakgrundsfärg anges blir lagret genomskinligt.

Dessutom kan du med attributet src=*"sökväg"* placera en hel separat hemsida i ett lager.

Exempel 17.4 a visar hur man skapar en rubrikskugga med hjälp av två lager. Figur 17.4 b visar resultatet.

Exempel 17.4 a
Med <layer>-koden skapar du lager utan att behöva använda style sheets. Z-index talar om vilket lager som ska ligga överst.

```
<layer left="20" top="5" z-index="2">  <! Första lagret>
<font face="sans-serif" size="7" color="#CCCCCC"><b>Lager
med vanlig HTML!</b></font>
</layer>
<layer left="23" top="8" z-index="1">  <! Andra lagret>
<font face="sans-serif" size="7" color="#000000"><b>Lager
med vanlig HTML!</b></font>
</layer>
```

Figur 17.4 b
Resultatet av kodningen i exempel 17.4 a. En rubrik med skugga.

Det första lagret ligger 20 pixlar från vänsterkanten och 5 pixlar från överkanten. Z-index="2" talar om att det ska ligga överst. Den övre rubriken visas med en gråfärgad (#CCCCCC) sans-serif av storlek 7.

Andra lagret är identiskt med det första, förutom att det placeras med tre pixlars förskjutning nedåt och till höger (left="23" top="8"). Z-index är 1 eftersom det här lagret ska ligga under det första. Också rubriken är identisk med den första, förutom att den är svart.

Lager i textrader med <ilayer>

Netscapes andra nyhet är <ilayer>…</ilayer> och gör det möjligt att skapa ett lager mitt inne i en textrad. Så här används koden:

```
Det här är en vanlig text men ett <ilayer left="3"
top="9">ord</ilayer> har flyttats.
```

Exempel 17.5 a och figur 17.5 b
Med <ilayer> kan man skapa ett lager mitt inne i en textrad.
left=*"värde"* och top=*"värde"* anger placeringen av objektet, i det här fallet ordet "ord", i förhållande till dess ursprungsläge.

Notera att left="3" och top="9" nu anger pixelposition i förhållande till det flyttade objektets *ursprungliga läge,* och inte, som i exempel 17.4 a, i förhållande till webläsarens ram. För övrigt fungerar attributen till <layer> oförändrade tillsammans med <ilayer>

Istället för z-index

Ett alternativ till att använda z-index är att namnge varje lager med attributet id=*"namn"* och sedan använta attributen above=*"lagernamn"* och below=*"lagernamn"* för att bestämma dess plats i djupled. Så här fungerar de tre attributen:

```
<layer id="lager1" above="lager2">Jag hamnar underst</layer>
<layer id="lager2" below="lager1">Jag hamnar i mitten</layer>
<layer id="lager3" below="lager2">Jag hamnar överst</layer>
```

Figur 17.6
Above och below ersätter z-index när du ska bestämma lagrets position i djupled.
För att bättre förstå det här lite bakvända systemet kan man tänka sig att below och above är en fråga, och lagernamnet ett svar. På första raden lyder till exempel frågan: "vilket lager ska ligga närmast ovanför det här lagret?" Svaret blir lager 2, vilket ger above="lager2".
Observera att z-index inte kan användas samtidigt som above och below.

Gardera med <nolayer>

Om du använder <layer>-koderna är det klokt att också ha med <nolayer> och </nolayer>. Allting som står mellan dessa båda koder visas bara i webläsare som inte förstår <layer> och <ilayer>

En varningstext kan se ut så här.

```
<nolayer>Den här sidan innehåller läckra lager. Ladda omedelbart
ned senaste versionen av Netscape Navigator!</nolayer>
```

Denna text kommer bara att kunna läsas av besökare som har fel webläsare. Det går naturligtvis också bra att lägga en komplett alternativ hemsida mellan <nolayer> och </nolayer>. Då kommer besökare med fel läsare aldrig att märka att de missade versionen med lager.

Koderna i sammanfattning

Viktigt.
Dessa koder fungerar inte med någon annan läsare än Netscape Navigator version 3.0 och 4.0.

<multicol>…</multicol>
Visar text i flera spalter.

Attribut till <multicol>

cols=*"värde"*
Anger antalet spalter.
gutter=*"värde"*
Anger avståndet mellan spalterna.
width=*"värde"* eller *"värde%"*
Anger textens bredd i pixlar eller procent av fönsterbredden.

<spacer>
Skapar ett tomrum.

Attribut till <spacer>

type=*"horizontal"*, *"vertical"* eller *"block"*
Anger tomrummets form.
size=*"värde"*
Anger storleken på en vertikal eller horisontell <spacer>.
width=*"värde"* och height=*"värde"*
Anger bredd respektive höjd för en <spacer> av blocktyp.
align=*"left"*, *"right"*
Anger justering för en <spacer> av blocktyp.

Viktigt.
<LAYER> och <ILAYER> fungerar bara med Netscape Navigator 4.0 eller senare versioner.

<layer>…</layer> och <ilayer>…</ilayer>
Skapar ett lager respektive ett lager i en textrad.

Attribut till <layer> och <ilayer>

id=*"lagernamn"*
Ger lagret ett namn.
left=*"pixelposition"* och top=*"pixelposition"*
Anger lagrets placering på sidan (<layer>) eller i relation till objektets normala placering (<ilayer>).
src=*"sökväg till html-dokument"*
Anger sökväg till dokument som ska visas i lagret.
z-index=*"värde"*
Anger lagrets placering i djupled.
above=*"lagernamn"* och below=*"lagernamn"*
Anger närmast över- och underliggande lager.
width=*"värde"* och height=*"värde"*
Anger lagrets dimensioner.
background=*"sökväg till fil"* och bgcolor=*"ärgkod"*
Anger lagrets bakgrundsbild respektive bakgrundsfärg.

KONSTEN ATT VISA SPECIALTECKEN:

18 Teckenkoder

World Wide Web talar tyvärr inte svenska. Det innebär vissa problem, särskilt för Macintosh-användare. I det här kapitlet får du lära dig varför datorer har problem med att visa svenska tecken, och hur du använder specialtecken så att din hemsida ändå blir läslig.

Alla som någon gång försökt flytta text mellan två olika typer av datorer, till exempel PC och Mac, vet att det är besvärligt. Resultatet blir oftast att svenska tecken försvinner och att radbrytningarna hamnar på fel ställen.

Orsaken är att datorer av olika märke inte alltid använder sig av samma system för att visa text. Man kan faktiskt säga att de talar olika språk.

I grunden använder visserligen alla datorer det så kallade ASCII-systemet (American Standard Code for Information Interchange). Det innebär att alla bokstäver och tecken tilldelats en siffra. Ett utropstecken har till exempel ASCII-koden 33 och bokstaven ”a” koden 65. När datorer skickar text till varandra är det egentligen ASCII-koderna för varje tecken som skickas. Mottagardatorn översätter sedan varje kod till rätt bokstav, varefter texten kan visas på skärmen.

För bokstäverna a–z och siffrorna 0–9 fungerar systemet utmärkt. Men datorerna PC, Macintosh och Unix använder inte samma ASCII-kod för den viktiga radbrytningen. Om du flyttar en engelsk text mellan dessa datorer visas texten därför oftast på rätt sätt, på det området förstår ju datorerna varandras ASCII-koder, medan radbrytningarna blir helt uppåt väggarna.

Ännu värre blir det om texten är på svenska. Våra svenska tecken å, ä och ö finns överhuvudtaget inte med i den ursprungliga tabellen över ASCII-koder. De fick inte plats.

Nu finns det en ny ASCII-tabell, så kallad *åttabitars ASCII*. Där finns

plats för 256 tecken, mot 128 i den gamla standarden, och därmed får nästan alla tecken som ingår i det latinska alfabetet plats. Typiskt nog hade dock både Apple och IBM hunnit hitta på egna varianter på åttabitars ASCII innan standarden slog igenom, vilket innebär att problemet med våra svenska specialtecken kvarstår i Mac- och MS-DOS-sammanhang.

Windows utvecklades däremot så sent att Microsoft hann anpassa systemet för den nya åttabitars teckentabellen. Ett märkligt resultat av detta är att Windows och MS-DOS inte använder en gemensam teckenstandard.

Windows och WWW talar samma språk

När World Wide Web konstruerades bestämde man, till alla Windows-användares glädje, att åttabitars ASCII skulle användas när HTML-sidor skickas över Internet. Detta innebär att du som jobbar med Windows kan skriva vilka tecken du vill i dina HTML-dokument, och lugnt räkna med att de visas på rätt sätt oavsett på vilken dator och med vilken webbläsare hemsidan läses. Exempel 18.1 a och figur 18.1 b visar hur Windows tolkar svenska tecken. Figur 18.1 c visar hur Macintosh tolkar samma sida. Eftersom Macintosh inte använder samma ASCII-tabell som Windows och World Wide Web blir tecknen felaktiga och texten svårläst.

Exempel 18.1 a
Om koden skrivs i Windows går det bra att använda vanliga svenska tecken.

```
<html>
<head><title>Titel</title><body>
<h1>Min hemsida</h1>
Det här är min första hemsida.<br />
Den är kort, men kärnfull.
</body></html>
```

Figur 18.1 b
Webbläsaren använder samma ASCII-tabell som Windows och de svenska tecknen visas därför på rätt sätt.

Figur 18.1 c
Om du skriver HTML med en Macintosh fungerar inte de svenska tecknen på hemsidan. Det beror på att Macintosh inte använder samma teckentabell som World Wide Web och Windows.

Lösningen: teckenkoder

Lyckligtvis byggde HTML:s konstruktörer in en lösning på problemet med de många teckentabellerna. Alla tecken tilldelades helt enkelt en särskild HTML-kod som alla webbläsare garanterat förstår.

Teckenkoder är inte uppbyggda som vanliga HTML-koder. De inleds alltid med ett et-tecken (&) och avslutas med ett semikolon (;). Däremellan står antingen en bokstavskombination som är tänkt att beskriva tecknet eller ett brädgårdstecken (#) följt av ett tal.

Så här ser koderna för våra svenska tecken ut:

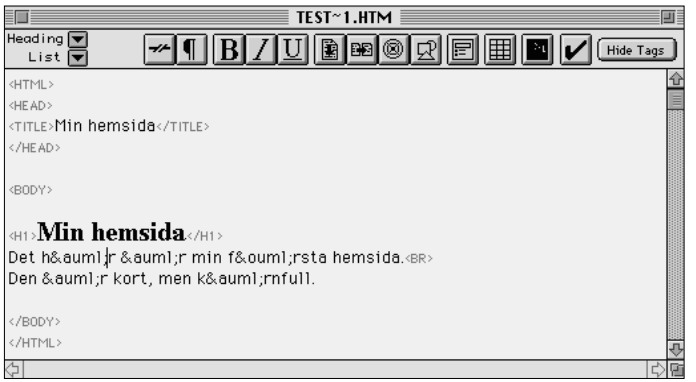
Tecken	Teckenkod	Sifferkod
å	&aring;	&#229;
ä	&auml;	&#228;
ö	&ouml;	&#246;
Å	&Aring;	&#197;
Ä	&Auml;	&#296;
Ö	&Ouml;	&#214;

Shareware. Shareware-systemet bygger på att du som användare får prova ett program under en viss tid, ofta 30 dagar. Om du sedan vill fortsätta att använda programmet måste du betala för det. Detta kallas ofta att *registrera* programmet hos tillverkaren.

Om du skriver HTML på en Macintosh måste du byta ut alla svenska tecken mot rätt ersättningskod. På en Mac måste alltså texten i exempel 18.1 a se ut så här för att de svenska tecknen ska fungera:

```
Det h&auml;r &auml;r min f&ouml;rsta hemsida.<br />
Den &auml;r kort, men k&auml;rnfull.
```

Om du inte vill skriva in koderna för hand, finns det HTML-editorer för Macintosh som automatiskt byter ut icke fungerande tecken mot rätt kod. En bra sådan editor är ”HTML Editor” (figur 18.2 på nästa sida) av amerikanen Rick Giles. Du hittar den på hans hemsida: http://dragon.acadiau.ca/~giles/. Programmet är **shareware** och är väl värt att registrera.



Figur 18.2
Programmet HTML Editor byter automatiskt ut alla svenska tecken och andra specialtecken mot koder som webläsarna förstår. Programmet är shareware och finns för nedladdning på adress http://dragon.acadiau.ca/~giles/

Andra specialtecken

Även du som använder Windows har användning för vissa teckenkoder. Om du till exempel vill skriva citationstecken (”), tecknen för mindre än och större än (< och >) eller et-tecknet (&) måste du använda teckenkoderna. Annars tror webbläsaren att tecknen ingår i själva HTML-koden, vilket kan röra till saker och ting rejält på din hemsida.

Det finns också specialtecken som åstadkommer användbara effekter.   ger till exempel ett så kallat *fast mellanslag,* som inte kan leda till en radbrytning. Detta kan du använda om du vill vara säker på att två ord, eller siffror, hamnar på samma rad. Ett fast mellanslag är också användbart då du behöver ett osynligt tecken att till exempel fylla en tabellcell med.

Med ett så kallat *mjukt bindestreck* () kan du tala om exakt var ett långt ord ska avstavas om det skulle behövas av utrymmesskäl.

Nedan finner du en lista över några användbara koder. I bilaga 3 finns en komplett förteckning över teckenkoderna som används i HTML.

Tecken	Teckenkod	Sifferkod
<	&lt;	&#060;
>	&gt;	&#062;
&	&amp;	&#038;
”	&quot;	&#034;
Fast mellanslag		
Mjukt bindestreck	&shy;	&#173;
©	&copy;	&#169;
®	&reg;	&#174;

KAN MAN VISA SIDORNA I MOBILEN?

19 XHTML

Även om din kod är "dålig" kommer förmodligen dina sidor att fungera och se ut ungefär som du tänkt dig, så varför göra det svårare än det är? Svaret ligger i att se framåt. Nästa steg i webbutvecklingen handlar till stor del om XML och metoden för att få HTML att fungera med XML är XHTML.

XHTML står för eXtensible HyperText Markup Language och är efterföljaren till HTML. Grunden till utvecklingen av XHTML är att fler och fler applikationer använder sig av XML (eXstensible Markup Language) som är ett strikt uppmärkningsspråk. Den grundläggande skillnaden mellan XML och HTML är att XML är designat för att *beskriva* data medan HTML är designat för att *visa* data. (Även om detta inte var det ursprungliga syftet med HTML.)

Idag finns en mängd olika sätt att visa data. Till exempel i en webbläsare på www, i en mobiltelefon, på en handdator eller varför inte på en kylskåpsdörr. De senare exemplen har inte möjlighet att tolka ett "dåligt" uppmärkt språk. Därför finns det anledning att använda metoder som gör att datan - sidans innehåll - kan hanteras på fler sätt än att visas i en webbläsare på www. Det är här XHTML kommer in och hjälper till att överbrygga skillnaderna och kommunikationsproblemen mellan de olika teknikerna. Det glädjande är att XHTML som en kombination av HTML och XML inte bara fungerar både idag och i morgon utan också igår...

Om du använder XHTML när du gör dina webbsidor kan dessa läsas av XML-applikationer men sidorna är dessutom "bakåt-kompatibla", de kommer alltså att fungera lika bra på gamla webbläsare som dina "gamla" sidor som är skrivna med HTML gjorde.

Viktiga skillnader mellan HTML och XHTML

- XHTML-element måste vara ordentligt nästlade
- XHTML-dokument måste vara välformade
- Namn på element måste skrivas med gemener (små bokstäver)
- Alla XHTML-element måste vara avslutade

Ordentligt nästlade element

Följande exempel visar skillnaden mellan korrekt och inkorrekt nästlad kod. Den "dåliga" koden fungerar dock tillfredsställande som HTML eftersom de flesta webbläsare tillåter det.

FEL: ****Denna text är kursiv och i fetstil****

För att koden ska vara korrekt XHTML måste taggar påbörjas och avslutas i rätt ordning, taggarna måste vara ordentligt *nästlade*. I det felaktiga exemplet påbörjas före men avslutas med före . Såhär ser det ut när koden är korrekt XHTML:

RÄTT: ****Denna text är kursiv och i fetstil****

Välformade dokument

Att ett dokument är välformat innebär att XHTML-elementen ligger innanför rot-elementet <html>. Alla element kan ha underelement så länge dessa är ordentligt nästlade. Såhär ser den grundläggande strukturen ut för ett välformat dokument:

```
<html>
<head>...</head>
<body>...</body>
</html>
```

Namn på taggar ska vara gemener

Namnen på taggarna ska vara gemener, små bokstäver. Detta beror på att XML-applikationer är känsliga för skillnader mellan gemener och versaler - små och stora bokstäver, <B> och tolkas alltså som två olika taggar. Först ett exempel på felaktig kod:

<P>Detta är ett nytt stycke med **<B>**fet**</B>** text.**</P>**

Såhär ska det se ut:

** **Detta är ett nytt stycke med ****fet**** text.** **

Alla element måste vara avslutade

De flesta taggar du använt i HTML har avslutningstaggar och även om du glömt skriva dem någon gång så har du nog märkt att det oftast fungerar ganska bra i alla fall. XHMTL är petigare med detta. Du behöver se till att alla taggar avslutas korrekt. Eftersom du är noggrann kommer du snart att upptäcka att det finns ett antal taggar som inte har avslutningstaggar i HTML. Ta till exempel taggen för en enkel radbrytning <br>, hur gör man med de taggar som inte har något avslut?

Här är exempel på tre taggar som saknar avslut i HTML:

```
Här kommer en radbrytning<br>
Här kommer en horisontell linje<hr>
Här kommer en bild<img src="bild.gif" alt="bild">
```

Lösningen på detta är att lägga in avslutningen inuti själva taggen. Ovanstående kod skriven som korrekt XHTML ser ut såhär:

```
Här kommer en radbrytning<br />
Här kommer en horisontell linje<hr />
Här kommer en bild<img src="bild.gif" alt="bild" />
```

Som du ser har ett snedstreck för avslutning lagts till i slutet av taggen. **Observera att det ska vara ett mellanslag innan snedstrecket**. Detta för att undvika kompabilitetsproblem med webbläsaren.

XHTML-syntax

Det finns några viktiga regler för hur syntaxen för XHTML ser ut och dessutom några skillnader mellan XHTML- och HTML-syntax. Dessa är de viktigaste reglerna och skillnaderna:

- Attributnamn ska skrivas med gemener
- Attributvärden måste skrivas mellan citattecken
- Attribut får inte "minimeras" eller förkortas
- Attributet "id" ersätter attributet "name"
- DTD:n för XHTML kräver vissa obligatoriska element

Det är inte så många regler och det är inte heller speciellt stor skillnad mot HTML-syntax. Den största skillnaden är att XHTML-syntaxen är mindre tillåtande. För att dina dokument ska uppfylla XHTML-standard **måste** koden följa anvisningarna, det räcker inte med att göra "nästan" rätt på det vis som du tidigare gjort med din HTML-kod. Visserligen kommer du fortfarande att kunna använda dina dina sidor som webbsidor men du kommer att stöta på problem om du vill använda innehållet i dina dokument i ett sammanhang där man arbetar med XML. Låter det onödigt krångligt? Misströsta inte, skillnaderna är små och ändringarna kommer att resultera i ett större användningsområde för dina dokument.

Attributnamn ska skrivas med gemener

Eftersom både namn på taggar och attribut ska skrivas med gemener kan du hålla utkik efter versaler när du kodar. Då är det lättare att hitta eventuella fel. Exemplet visar felaktigt och korrekt sätt att skriva attrubutnamn:

FEL:
```
<table WIDTH="100%">
<tr>
<td ALIGN="right">Celltext</td>
</tr>
</table>
```

RÄTT:
```
<table width="100%">
<tr>
<td align="right">Celltext</td>
</tr>
</table>
```

Attributvärden mellan citattecken

Alla attributvärden ska stå mellan citattecken. Du kan använda enkla eller dubbla citattecken så länge du är konsekvent, alltså till exempel inte börjar med ett dubbelt och slutar med ett enkelt. Det vanligaste är dock att man använder dubbla citattecken.

FEL:
```
<table width=100%><tr>
<td align=right>Celltext</td>
</tr></table>
```

RÄTT:
```
<table width="100%"><tr>
<td align="right">Celltext</td>
</tr></table>
```

Ingen "minimering" av attribut

På vissa ställen i din HTML-kod har du skrivit "minimerade" attribut. Ett exempel på detta är attributet **selected** i nedanstående kod.

```
<select id="valjare">
<option value="vald" selected>Vald</option>
</select>
```

För att syntaxen ska vara korrekt ska det se ut såhär:

```
<select id="valjare">
<option value="vald" selected="selected">Vald</option>
</select>
```

Här är en lista över de attribut som är minimerade i HTML och som ska skrivas ut i XHTML:

HMTL	XHTML
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled ="disabled "
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noresize	noresize="noresize"
noshade	noshade="noshade"
nowrap	nowrap="nowrap "
multiple	multiple="multiple"

Attributet *id* ersätter attributet *name*

Du har förmodligen hittills använt attributet *name* till en mängd taggar. Enligt XHTML-syntaxen ska attributet *name* inte finnas utan i stället ersättas av attributet *id*. Här bör du vara lite försiktig. Om du till exempel använder javascript som refererar till objekt, namngivna med *name* kan en ändring av *name* till *id* innebära att det uppstår fel. Det bästa sättet att undvika fel av den här typen och samtidigt försäkra dig om att även gamla webbläsare kan använda dina sidor är använda **både *name* och *id***. De element som berörs av denna ändring är **<a>**, **<applet>**, **<frame>**, **<iframe>**, **<img>** och **<map>**.

FEL:

RÄTT:

RÄTT:

Obligatoriska element i XHTML

De element som är obligatoriska i XHTML är DOCTYPE, <html>, <head>, och <body>. Elementet <title> ska finnas inne i <head>-elementet. Elementet DOCTYPE anger vilken typ av innehåll dokumentet har och hur det ska tolkas, mer om DOCTYPE och DTD lite senare.

Ett XHTML-dokument med obligatoriska element kan alltså se ut såhär:

```
<!DOCTYPE här finns dokumenttypen>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Sidans titel</title>
</head>
<body>
Sidans innehåll finns här
</body>
</html>
```

Som du ser finns det ett attribut till <html>. Detta är visserligen obligatoriskt men det är också ett statiskt värde som kommer att tolkas på samma sätt, oavsett om du har det med eller inte. Notera att DOCTYPE-elementet sitter allra först, alltså inte inom <html>-taggarna. Detta för att DOCTYPE inte är html utan anger vilken DTD som ska användas. DOCTYPE-elementet har inte heller någon avslutning.

Nu har vi de element som behövs, det enda som saknas är angivelser för vilken typ av dokument vi vill göra. Vad har sidan för innehåll och hur vill vi att den ska tolkas. Vi ska nu bestämma detta genom att ange en av de tre DTD:er som finns för XHTML.

DTD och DOCTYPE

DTD Står för Document Type Definition och specificerar syntaxen för en en webbsida i SGML. DTD används av SGML-applikationer, som till exempel HTML, för att specificera regler som är knutna till hur man märker upp ett dokument av en speciell sort. DTD:n inkluderar ett uppsättning element och regler.

XHTML specificeras av en SGML-DTD och denna XHTML-DTD beskriver på ett precist, dator-läsbart sätt, den tillåtna syntaxen och grammatiken för XHTML. Låter det krångligt? Lugn, i XHTML 1.0 finns bara tre olika DTD:er att välja mellan; Strict, Transitional och Frameset.

Tre olika DOCTYPE, DTD:er

Den första typen heter *Strict* och som namnet antyder ska man använda denna när koden är "strikt" eller "ren", alltså utan några designkoder och HTML-taggar som anger *hur* innehållet ska presenteras. Du kan fortfarande styra sidans design men om du använder *Strict* ska du använda CSS - stylesheets - för designen av sidan. Såhär ser DOCTYPE-elementet ut för Strict:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Som du ser följer inte attributen i detta element de syntaxregler för XHTML som vi gått igenom vilket kan vara förvirrande men tänk på att elementet inte är en del av själva XHTML-dokumentet och inte heller ett XHTML-element.

Nästa typ av DOCTYPE är *transitional*. Denna ska du använda när du vill dra nytta av tillgångarna med HTML. Ett annat tillfälle att använda *transitional* är när du vill göra din sida tillgänglig för webbläsare som inte hanterar stylesheets. Du ser att skillnaden från *Strict*-koden är liten.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Den sista DOCTYPEN är *Frameset* och det säger sig självt att du ska använda detta alternativ när ditt dokument är ett frameset-dokument. Det betyder inte att du ska använda *Frameset* för **alla** de sidor som ingår i ditt frameset utan bara för själva frameset-dokumentet. Och skillnaden mot de tidigare? Det har du redan listat ut. Så här ska det se ut när DOCTYPE är *Frameset.*

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Ett dokument som innehåller de obligatoriska och grundläggande elementen för XHTML kan alltså se ut såhär:

```
1  <!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <title>Ett dokument</title>
5  </head>
6  <body>
7  <p>Ett stycke</p>
8  </body>
9  </html>
```

Koden rad för rad:

1. Här anges att dokumetet ska tolkas enligt principerna för XHTML-DTD:n *Transitional*.
2. Den vanliga <html>-taggen tillsammans med ett attribut för XML Namespace Catalog. Detta attribut kan uteslutas men om det inte finns med kommer en validering förmodligen att notera detta.
3. <head>-taggen för dokumentet.
4. Dokumentets titel omsluten av <title> och </title>.
5. Avslutning av dokumentets header med </head>.
6. Start för dokumentets "kropp", <body>.
7. En rad med ett stycke, startar med och avslutas med .
8. Avslutning av dokumentets "kropp" med </body>.
9. Avslutning av html med </html>.

Vissa editorer kan redan skriva eller XHTML-kod eller konvertera HTML-kod till XHTML-kod, t.ex. programmen Dreamweaver MX 2004 och HTML-Kit. Det finns också en mängd andra hjälpprogram som är mer eller mindre lätta att använda för att kontrollera syntaxen och konvertera HTML till XHTML. Om du är PC-användare och inte är främmande för att använda DOS-prompten kan du t.ex. ladda ner programmet HTML-tidy från **http://tidy.sourceforge.net/**.

När dina sidor är ute på webben kan du kontrollera att sidorna fungerar som giltig XHML med hjälp av en validering. Valideringen går igenom sidornas kod och noterar vad som behöver åtgärdas för att koden ska bli korrekt XHTML. På webbplatsen **http://validator.w3.org/** kan du köra dina sidor mot W3C:s egen validering.

HTML-referens

Den här referensen innehåller alla de HTML-koder du kan få användning för när du gör hemsidor. Om inget annat anges fungerar koderna med Internet Explorer version 2.0 och senare och Netscape Navigator 2.0 och senare. Koder som bara fungerar med en särskild läsare eller en särskild version markeras med läsarens initialer och versionsnummer, till exempel ”ie4·ns2-4”.

HTML-kod

<!- - kommentartext - ->
Kommentar. Allt som placeras mellan ”<!- -” och ”- ->” och som inte är HTML-koder ignoreras av webläsaren.

<!DOCTYPE>
Placeras överst i dokumentet och anger version av HTML. Ingår i HTML 3.2.

<a>...
Skapar en länk eller ett ankare.

Attribut till <a>

base target=*”fönsternamn”*
Anger ett fönster dit alla länkar skickas. ie3-4·ns2-4

href=*”sökväg#ankarnamn”*
Skapar en länk till ett annat dokument och/eller ankare.

mailto:*name@postadress*
Skapar en länk till ett föradresserat brev.

name=*”ankarnamn”*
Skapar ett ankare.

target=*”fönsternamn”*
Laddar länken i det namngivna fönstret.

title=*”titel”*
Anger namn på fönstret.

<abbr>...</abbr>
Anger kortform.

Attribut till <abbr>

title=*”kortformen utskriven”*

<acronym>...</acronym>
Anger initialord som uttalas som ett ord, t.ex NATO

Attribut till <acronym>

title=*”akronymen utskriven”*

<address>...</address>
Anger adress. Oftast samma effekt som ….

<applet>...</applet>
Lägger in en Java applet i dokumentet. OBS! fungerar ej när XHMTL DOCTYPE är *Strict*!

Attribut till <applet>

align=*”left”*, *”center”* eller *”right”*
Anger horisontell justering.

alt=*”alternativ text”*
Text som visas om läsaren inte stöder Java.

archive=*”sökväg”*
Anger sökväg när applet är sparad som Java-arkiv eller zip-fil.

code=*”applet”*
Namn på aktuell Java applet.

codebase=*”sökväg”*
Sökvägen till katalogen där appleten finns.

height=*”värde”*
Appletens höjd i pixlar.

hspace=*”värde”*
Horisontell yttermarginal i pixlar.

name=*”namn”*
Ger appleten ett namn.

param name=*”attributnamn”*
Används för att skicka kommandon från HTML-sidan till appleten.

vspace=*”värde”*
Vertikal yttermarginal i pixlar.

width=*”värde”*
Appletens bredd i pixlar.

<area />
Definierar ett klickbart område i en klientbaserad bildkarta (<map>).

Attribut till <area />

alt =*”alternativ text”*
Anger en alternativ text för arean.

coords=*”koordinater”*
Anger områdets koordinater enligt följande:

Rektangel: *x1,y1,x2,y2*
Cirkel: *x,y,r* där *x,y* är cirkelns mittpunkt och *r* radien.
Polygon: *x1,y1,x2,y2,x3,y3* osv
href=*"sökväg"*
Anger vilken länk som ska vara kopplad till området.
nohref
Ersätter href=*"sökväg"* och gör så att ingenting händer om användaren klickar i området.
shape=*"form"*
Anger områdets form. Kan vara *rect* (rektangel), *poly* (polygon), *circle* (cirkel) eller *default*.
target=*"fönsternamn"*
Anger vart länken ska skickas. ie2-4

…
Skapar fet stil. ie2-4·ns4

<base />
Anger ett dokuments hemadress eller generell fönsteradress för länkar. Ska finnas i dokumentets <head>. ie2-4
Attribut till <base />
href=*"sökväg"*
Anger dokumentets hemadress. Alla relativa länkar utgår sedan från denna adress.
target=*"window"*
Anger ett fönster dit alla länkar skickas.

<basefont />
Anger grundinställningar för text. ns2-4
Attribut till <basefont>
size=*"värde"*
Anger en grundstorlek för text.
color=*"#färgkod"*
Anger en grundfärg för text.
face=*"alternativ1, alternativ2* osv*"*
Anger lista på alternativa grundtypsnitt.

<bdo>…</bdo> ie4·ns2-4
Anger riktning för text
Attribut till <bdo>
dir=*"ltr"* (left to right) eller *"rtl"* (right to left)

<bgsound />
Anger ett bakgrundsljud.
Attribut till <bgsound /> ie2-4
src=*"sökväg"*
Anger sökväg till ljudfilen. Ljudet kan vara i WAV, AU eller MIDI-format. ie2-4
loop=*"värde"* eller *"infinite"*
Anger antal uppspelningar.

<big>…</big>
Gör text en storlek större.

<blink>…</blink> ie2-4
Gör så att texten blinkar.

<blockquote>…</blockquote>
Skapar ett indrag av både höger och vänster marginal.

<body>…</body>
Skapar HTML-dokumentets kroppsdel.
Attribut till <body>
alink=*"#färgkod"*
Anger färgen på en aktiv länk.
background=*"sökväg"*
Anger bakgrundsbild.
bgcolor=*"#färgkod"*
Anger bakgrundsfärg på hemsidan.
bgproperties=*"fixed"*
Gör bakgrundsbilden stillastående vid scroll. ie2-4
leftmargin=*"värde"*
Anger en vänstermarginal i pixlar. ie2-4
link=*"#färgkod"*
Anger färgen för icke besökta länkar.
text=*"#färgkod"*
Anger hemsidans textfärg.
topmargin=*"värde"*
Skapar en övre marginal. Mäts i pixlar.
vlink=*"#färgkod"*
Anger färgen på besökta länkar. ie3-4

Skapar en radbrytning.
**Attribut till
**
clear=*"left", "right"* eller *"all"*
Bryter raden så att den hamnar under en vänsterställd flytande bild, högerställd flytande bild eller under den lägsta flytande bilden.

<caption>…</caption> ie3-4
Skapar en rubrik till en tabell.
Attribut till <caption>
align=*"center", "left"* eller *"right"*
Anger rubrikens horisontella placering.
valign=*"top"* eller *"bottom"*
Anger rubrikens vertikala placering.

…
Centrerar text och bilder.

<cite>…</cite>
Logiskt citat. Samma effekt som ….

<code>…</code>
Logiskt kodexempel. Effekt som <tt>.

<col>…</col>
Styr egenskaperna hos enskilda kolumner i en kolumngrupp (skapas med <colgroup>). ie3-4·ns2-4
Attribut till <col>
span=*"värde"*
Anger antal kolumner som berörs.
align=*"left"*, *"center"* eller *"right"*
Anger justering av innehållet i kolumnens celler. ie3-4·ns4

<colgroup>…</colgroup>
Skapar en grupp av kolumner.
Attribut till <colgroup> ie3-4·ns4
span=*"värde"*
Anger antal kolumner gruppen omfattar. ie3-4·ns4
align=*"left", "center"* eller *"right"*
Anger justering av innehållet i kolumnernas celler.

<dd>…</dd>
Markerar ett definitionsord i en <dl>.

~~…~~ (används med <ins>)
Anger text som tagits bort ur dokumentet.
Text inom <del> blir överstruken.
Attribut till <del> (valfria)
cite=*"sökväg till dokument med förklaring"* ie3-4·ns2-4
datetime=*"åååå-mm-dd"*

<dfn>…</dfn>
Logisk definition. Samma effekt som .

<dir>…</dir>
Skapar en kataloglista. Fungerar som en onumrerad lista (<ul>).

…
Tilldelar mellanliggande text egenskaper enligt attributen. Används med style sheets.
**Attribut till **
align=*"left", "right"* eller *"center"*
Justerar texten horisontellt.
style=*"attribut1: värde1;* osv*"*
Anger det lokala style sheet som ska appliceras på texten.
id=*"namn"*
Knyter koden till ett id.
class=*"namn"*
Knyter koden till en klass.

<dl>…</dl>
Skapar en definitionslista. Listan innehåller <dt> och <dd>.

<dt> …</dt>
Föregår ordet som ska definieras i en definitionslista.

…
Logiskt eftertryck. Effekt som ….

<embed>…</embed>
Inkluderar t ex en Java applet i dokumentet.
Attribut till <embed>
height=*"värde"*
Anger objektets höjd.
name=*"namn"*
Ger objektet ett namn.
palette=*"foreground"* eller *"background"* ie3-4·ns2-4
Anger typ av färgkarta.
src=*"sökväg"*
Anger sökväg till objektet.
width=*"värde"*

<fieldset>…</fieldset>
Ritar en ruta runt de element som taggen omsluter.

…
Ger egenskaper till text.
Attribut till <font>
size=*"värde"* eller *"+/–värde"*
Sätter en storlek på efterföljande text. Värdet kan vara 1 (minst) till 7 (störst). +/– värde ändrar storlek i steg i förhållande till basefont-värdet.
color=*"#färgkod"*
Anger textfärg.
face=*"alternativ1, alternativ2,* osv" ie3-4·ns2-4
Anger typsnitt.

<form>…</form>
Skapar ett formulär.
Attribut till <form>
action=*"adress"*
Anger vart formulärets innehåll ska skickas.
method=*"get"* eller *"post"*
Anger hur innehållet ska skickas.
name=*"namn"*
Ger formuläret ett namn.
target=*"fönsternamn"*
Anger vilket fönster resultatet av formuläret ska laddas i.

<frame />
Skapar en ruta.
Attribut till <frame />
src=*"sökväg"*
Anger HTML-dokument som ska visas i rutan.
name=*"namn"*
Ger rutan ett namn.

marginwidth=*"värde"*
Skapar en vänster- och högermarginal mellan rutans innehåll och ramen.
marginheight=*"värde"*
Skapar en övre och undre marginal.
frameborder=*"0"* eller *"1"*
Anger om rutan ska ha ram eller ej.
scrolling=*"yes"*, *"no"* eller *"auto"* ie3-4
Avgör om rutan ska ha en rullist eller inte.
noresize
Hindrar att ramarna flyttas.
longdesc=*"sökväg"*
Sökväg till beskrivande text om framens innehåll, använd för webbläsare som ej hanterar frames.

<frameset>…</frameset>
Skapar ett rutsystem.
Attribut till <frameset>
rows=*"värde"*, *"värde%"* eller *"värde*"*
Anger antalet rader, och deras höjd. ie3-4
cols=*"värde"*, *"värde%"* eller *"värde*"*
Anger antalet kolumner, och deras höjd.
frameborder=*"0"* eller *"1"*
Slår på och av ramen i hela rutsystemet.
framespacing=*"värde"*
Anger avstånd mellan rutorna. OBS bara ie.
border=*"värde"*
Anger avstånd mellan rutorna. OBS bara ns.

<h*värde*>…</h*värde*>
Skapar en rubrik. Värdet kan vara mellan 1 (störst) och 6 (minst).
Attribut till <h*värde*>
align=*"left"*, *"center"* eller *"right"*
Justerar rubriken horisontellt.

<head>…</head>
Skapar dokumentets huvud. Följande element kan finnas inom sidhuvudet:
<base /><link /><meta />
<script><style><title>

<hr />
Skapar en horisontell linje.
Attribut till <hr /> ns4
align=*"left"*, *"right"* eller *"center"*
Anger linjens justering.
noshade
Tar bort linjens skugga.
color=*"#färgkod"*
Anger linjens färg.
size=*"värde"*
Anger linjens tjocklek i pixlar.
width=*"värde"* eller *"värde%"*
Anger linjens bredd i pixlar eller procent.

<html>…</html>
Skapar hela HTML-dokumentet.

…
Gör texten kursiv.

<iframe>…</iframe>
Skapar en flytande ruta.
Attribut till <iframe>
align=*"top"*, *"center"*, *"middle"*, *"left"* eller *"right"*
Anger rutans justering.
frameborder=*"1"* eller *"0"*
Slår på eller av rutans ram.
height=*"värde"*
Anger rutans höjd i pixlar.
width=*"värde"*
Anger rutans bredd i pixlar.
marginheight=*"värde"* ie3-4
Anger vertikal marginal i pixlar.
marginwidth=*"värde"*
Anger horisontell marginal i pixlar.
name=*"namn"*
Ger rutan ett namn så att den kan ta emot hyperlänkar.
scrolling=*"yes"* eller *"no"*
Slår på och av rullisten.
src=*"sökväg"*
Anger HTML-dokument som ska visas i rutan.
longdesc=*"sökväg"*
Sökväg till beskrivande text om framens innehåll.

<ilayer>…</ilayer> ie2-4
Skapar ett lager i en textrad.
Attribut till <ilayer>
id=*"lagernamn"*
Ger lagret ett namn.
left=*"pixelposition"*
Anger lagrets placering horisontellt i förhållande till objektets ursprungsläge.
top=*"pixelposition"* ns4
Anger lagrets placering vertikalt i förhållande till objektets ursprungsläge.
src=*"sökväg"*
Sökväg till dokument som ska visas i lagret.
z-index=*"värde"*
Anger lagrets placering i djupled.
above=*"lagernamn"*
Anger närmast överliggande lager. ie2-4
below=*"lagernamn"*
Anger närmast underliggande lager.
width=*"värde"*
Anger lagrets bredd. ie2-4
height=*"värde"*
Anger lagrets höjd.
background=*"sökväg"*
Anger lagrets bakgrundsbild
bgcolor=*"färgkod"*
Anger lagrets bakgrundsfärg.

<img />
Placerar en bild på hemsidan.
Attribut till <img />
align=*"top"*, *"middle"*, *"bottom"*, *"left"* eller *"right"*
Anger bildens justering i förhållande till omgivande text.
alt=*"alternativ text"*
Anger en alternativ text.
border=*"värde"*
Anger hur bred bildens ram ska vara.
Fungerar bara med klickbara bilder i IE.
controls
Visar kontroller om bilden är ett videoklipp.
dynsrc=*"sökväg"*
Anger sökväg till en VRML-värld eller ett videoklipp. Betyder "Dynamic Source".
height=*"värde"*
Anger höjd. Mosaic ändrar inte storleken.
hspace=*"värde"*
Anger horisontell marginal.
src=*"sökväg"*
Anger sökvägen till bilden.
vspace=*"värde"*
Anger vertikal marginal.
width=*"värde"*
Anger bredd. Mosaic ändrar inte storleken.
ismap
Anger att bilden är en serverbaserad bildkarta.
loop=*"värde"*
Definierar hur många gånger ett videoklipp ska visas. Om LOOP=*–1* eller LOOP=*infinite* körs det i en ändlös loop. ns2-4
start=*"startpunkt"*
Anger när filen som specificerats med dynsrc ska börja visas. Startpunkten kan vara *"fileopen"*, vilket startar filen så fort den laddats ned, eller *"mouseover"*, som startar uppspelningen när pekaren är över bilden.
usemap=*"karta"*
Anges i en klientbaserad bildkarta.
longdesc=*"sökväg"* ie2-4
Sökväg till beskrivande text om bilden.

<input />
Skapar ett formulärobjekt.
Attribut till <input />
checked
Gör en kryssruta eller radioknapp förvald.
disabled
Gör att ett formulärobjekt inte är aktiverat.
maxlength=*"värde"*
Anger maximalt antal tecken som får skrivas i en textruta.
name=*"namn"*
Ger formulärobjektet ett namn.
readonly
Gör att ett fomulärobjekt inte är editerbart.
size=*"värde"*
Anger en textrutas längd mätt i antal tecken.
src=*"sökväg"*
Anger sökväg till bild då type=*"image"*.
type=*"checkbox"*, *"radio"*, *"text"*, *"password"*, *"hidden"*, *"submit"*, *"reset"*, *"button"* eller *"image"*
Anger typ av <input>.
value=*"värde"* ie3-4·ns4
Ger formulärobjektet ett värde.

<ins>…</ins> (används med <del>)
Anger infogad text.
Attribut till <ins>
cite=*"sökväg"*
Anger sökväg till förklarande text
datetime=*"åååå-mm-dd"*
Anger tidpunkt för infogningen.

<isindex>
Meddelar att ett system av hemsidor är sökbart. Kräver CGI-program.
Attribut till <isindex>
action=*"sökväg"*
Anger vart söksträngen ska sändas.
prompt=*"text"*
Anger instruktionstext som ska visas.

<kbd>…</kbd>
Logisk tangentbordsinmatning. Effekt som <b> kombinerat med <tt>.

<label>…</label>

Används för att binda text till ett formulärobjekt.

Attribut till <label>

for=*"id för formulärobjekt"*

<legend>...<legend>

Anger en rubrik för ett <fieldset>.

<layer>...</layer>

Skapar ett lager.

Attribut till <layer>

id=*"lagernamn"*
Ger lagret ett namn.
left=*"pixelposition"*
Anger lagrets placering horisontellt.
top=*"pixelposition"* ie2-4
Anger lagrets placering vertikalt.
src=*"sökväg"*
Anger sökväg till dokument som ska visas i lagret.
z-index=*"värde"*
Anger lagrets placering i djupled.
above=*"lagernamn"*
Anger närmast överliggande lager.
below=*"lagernamn"*
Anger närmast underliggande lager.
width=*"värde"*
Anger lagrets bredd.
height=*"värde"*
Anger lagrets höjd.
background=*"sökväg"*
Anger lagrets bakgrundsbild.
bgcolor=*"#färgkod"*
Anger lagrets bakgrundsfärg.

...

Definierar en rad i en onumrerad (<ul>), numrerad (<ol>) kataloglista (<dir>) eller menylista (<menu>).

**Attribut till **

type=*"A", "a", "I", "i"* eller *"1"*
Anger hur en numrerad lista ska numreras.
value=*"värde"*
Används för att göra ett hopp i listan.

<link />

Anger en relation till ett annat dokument. Kan t ex anropa ett länkat style sheet eller ett dynamiskt typsnitt.

Attribut till <link />

charset=*"charset"* (standard är ISO-8859-1)
href=*"sökväg"*
Anger sökvägen till dokumentet.
hreflang=*"språkkod"*
Anger basspråk för måldokumentet.
media=*"mediatyp"*
Anger med vilken mediatyp dokumentet ska användas, t.ex. *all, braille* eller *print*.
rel=*"länktyp"*
Anger typ av länkdokument, t ex "stylesheet" eller "fontdef".
rev=*"länktyp"*
Anger typ av länkdokument bakåt. ns3-4
title=*"title"*
Anger rekommenderad titel.
type=*"dokumenttyp"*
Anger MIME-typ, t ex "text/css".
src=*"adress"*

<map>...</map>

Skapar en kartfil för klientbaserad bildkarta.

Attribut till <map>

name=*"kartnamn"*
Ger kartan ett namn.
id=*"kartid"*
Ger kartan ett id.

Syntax för kartfiler:

CERN-modellen
rect (x1,y1) (x2,y2) länkadress
poly (x1,y1) (x2,y2) (x3,y3) osv ie3-4
länkadress
circle (x1,y1) r länkadress
NCSA-modellen
rect länkadress x1,y1 x2,y2
poly länkadress x1,y1 x2,y2 x3,y3 osv
circle länkadress x1,y1 x2,y2

<marquee>...</marquee>

Skapar en ruta med rullande text.

Attribut till <marquee>

align=*"left", "center"* eller *"right"*
Anger rutans horisontella justering.
behavior=*"scroll", "slide"* eller *"alternate"*
Anger textens sätt att röra sig.
bgcolor=*"#färgkod"*
Anger textens bakgrundsfärg.
direction=*"left"* eller *"right"*
Anger textens rörelseriktning.
height=*"värde"* eller *"värde%"*
Anger rutans höjd i pixlar eller procent av fönstrets höjd.
width=*"värde"* eller *"värde%"*
Anger rutans bredd i pixlar eller procent av fönstrets bredd.
hspace=*"värde"*
Anger horisontell marginal utanför rutan.
vspace=*"värde"*
Anger vertikal marginal utanför rutan.
loop=*"värde"*
Anger antalet loopar texten ska göra.
scrollamount=*"värde"*
Anger antal pixlar per steg.
scrolldelay=*"värde"*
Anger antal millisekunder mellan varje steg.

<menu>...</menu>

Skapar en menylista. Fungerar som onumrerad lista (<ol>).

<meta />

Förser webläsaren med info om dokumentet.

Attribut till <meta />

http-equiv=*"åtgärd"*
Får webläsaren att utföra en åtgärd.
content=*"innehåll"*
Definierar meta-information för åtgärden.
name=*"beskrivning"*
Ger en beskrivning av dokumentet.
url=*"sökväg"*
Anger dokumentets sökväg.

Exempel på <meta /> i aktion:

<meta http-equiv="refresh" content="5; URL=sökväg" /> ger ett automatiskt hopp till HTML-dokumentet i *sökväg* efter 5 sekunder.

<multicol>...</multicol>

Visar text i flera spalter.

Attribut till <multicol>

cols=*"värde"*
Anger antalet spalter.
gutter=*"värde"*
Anger avståndet mellan spalterna. ie3-4
width=*"värde"* eller *"värde%"*
Anger textens bredd i pixlar eller procent av fönsterbredden.

<nobr>...</nobr>

Stoppar alla radbrytningar.

<noframes>...</noframes>

Den inneslutna koden läses bara av webläsare som inte klarar rutsystem. Används i frameset-dokument.

<noscript>...</noscript>

Innesluten text visas när webbläsaren inte kan hantera scripttypen.

<object>...</object>

Inkluderar t ex Java applet i dokumentet.
Ersätter <embed>.

Attribut till <object>

align=*"baseline", "center", "left", "middle", "right", "textbottom", "textmiddle"* eller "*texttop*"
Anger objektets justering.
border=*"värde"*
Definierar ramen. ie3-4·ns2-4
classid=*"sökväg"*
Identifierar objektet.
codebse=*"sökväg"*
Anger koden för objektet.
codetype=*"kodtyp"*
Anger typ av kod.
data=*"sökväg"*
Ger data till objektet.
declare
Annonserar objektet för kommade anrop.
height=*"värde"*
Anger objektets höjd.
hspace="*värde*"
Objektets yttre horisontella marginal.
name=*"namn"*
Namnger objektet när det skickas som del i ett formulär.
shapes ns3-4
Anger att objektet innehåller klickbara fält.
standby=*"text"*
Anger text som visas under laddning av objektet.
type=*"dokumenttyp"*
Anger MIME-typ.
usemap=*"sökväg"*
Anger sökväg till en kartfil.
vspace=*"värde"*
Anger objektets vertikala yttre marginal.
width=*"värde"*
Anger objektets bredd.

... ie3-4·ns4

Skapar en numrerad lista. Varje rad inleds med <li>.

**Attribut till **

type=*"A", "a", "I", "i"* eller "*1*"
Anger hur listan ska numreras.

<optgroup>...</optgroup>

Grupperar flera <option> i en <select>-lista.

Attribut till <optgroup>

label=*"rubriknamn"*
Anger rubrik för grupperingen.

<option>...</option>
Skapar ett val i en <select>-meny i ett formulär.
Attribut till <option>
selected=*"selected"* ie3-4·ns4
Gör alternativet förvalt.
value=*"värde"*
Ger alternativet ett värde.
...
Skapar nytt stycke.
**Attribut till **
align=*"left"*, *"right"* eller *"center"*
Anger textens justering.
<param />
Specificerar parametrar för ett objekt.
Attribut till <param />
name=*"namn"*
Ger parametrarna ett namn.
value=*"värde"*
Anger värde på parametrarna.
valuetype=*"data"*, *"ref"* eller *"object"*
Anger typ av värde.
type=*"typ"*
Anger MIME-typ.
<plaintext>...</plaintext>
Stänger av HTML-kodernas funktioner och visar koderna.
<pre>...</pre>
Visar text med de radmatningar och tabulatorer som ingår i HTML-dokumentet. Visas med fast bredd.
<s>...</s>
Skapar genomstruken text.
<samp>...</samp> ie3-4
Logiskt textexempel. Effekt som <tt>.
<script>...</script>
Anger att dokumentet innehåller script-kod. ie3-4
Attribut till <script>
language=*"scriptspråk"*
Anger språk, t ex JavaScript. ie3-4
type=*"typ"*
Anger MIME-typ. ie3-4
src=*"sökväg"*
Anger adress till fil som innehåller scriptet. ie3-4
<select>...</select>
Skapar en meny i ett formulär. Varje alternativ anges med <object>. ie3-4
Attribut till <select> ie3-4
disabled="disabled"
Anger att menyn ska vara avaktiverad.
multiple="multiple"
Tillåter att flera alternativ väljs.
name=*"namn"*
Ger menyn ett namn.
size=*"värde"*
Anger menyns längd. size="1" ger popup-meny.
<small>...</small>
Gör text en storlek mindre.
<spacer />
Skapar ett tomrum.
Attribut till <spacer />
type=*"horizontal"*, *"vertical"* eller *"block"*
Anger tomrummets form.
size=*"värde"* ie2-4·ns3-4
Anger storlek (vertikal eller horisontell typ).
width=*"värde"* ie3-4·ns4
Anger bredd (vid blocktyp).
height=*"värde"* ie3-4
Anger höjd (vid blocktyp).
align=*"left"* eller *"right"* ie3-4
Anger justering (vid blocktyp).
... ie3-4
Applicerar style-information på ett avsnitt.
**Attribut till **
style=*"attribut1: värde1;* (osv)*"*
Anger det lokala style sheet som ska appliceras på texten.
id=*"namn"*
Knyter koden till ett id.
class=*"namn"*
Knyter koden till en klass.
<strike>...</strike>
Skapar genomstruken text.
...
Logiskt eftertryck. Effekt som ....
<style>...</style>
Definierar ett globalt style sheet. Placeras i <head>-koden.
Attribut till <style>
type=*"text/css"* ie3-4
Talar om att koden som följer är ett style sheet.
media=*"mediatyp"*
Anger typ av media t.ex. *all, handheld, braille*
_{...}
Skapar mindre, nedsänkt text. ie3-4
^{...}

Skapar mindre, upphöjd text.
<table>...</table>
Skapar en tabell.
Attribut till <table>
border=*"värde"*
Anger rambredd.
cellspacing=*"värde"*
Anger avståndet mellan tabellens celler.
cellpadding=*"värde"*
Anger avståndet mellan ram och cellinnehåll.
width=*"värde"* eller *"värde%"* ie2-4·ns3-4
Anger tabellens bredd.
height=*"värde"* eller *"värde%"* ie3-4
Anger tabellens höjd.
align=*"left"* eller *"right"* ie3-4
Får hela tabellen att flyta till vänster eller höger. ie3-4
bgcolor=*"#färgkod"*
Anger tabellens bakgrundsfärg.
rules=*"none"*, *"groups"*, *"rows"*, *"cols"* eller *"all"*
Anger del av tabellen som ska skiljas åt med linjer.
frame=*"void"*, *"above"*, *"below"*, *"hsides"*, *"vsides"*, *"lhs"*, *"rhs"* eller *"box"*
Anger hur tabellens yttre ram ska se ut.
background=*"sökväg"*
Anger bakgrundsbild till tabellen.
bordercolor=*"#färgkod"*
Anger ramfärg.
bordercolordark=*"#färgkod"*
Anger den mörka färgen i en 3D-ram.
bordercolorlight=*"#färgkod"*
Anger den ljusa färgen i en 3D-ram.
summary=*"text"*
Anger en summering av tabellinnehållet, avsett för icke-visuella webbläsare, t.ex. talsyntes.
<tbody>...</tbody>
Skapar en tabellkropp.
<td>...</td>
Skapar en cell i en tabell.
Attribut till <td>
align=*"left"*, *"center"* eller *"right"*
Anger cellinnehållets horisontella justering.
valign=*"top"*, *"middle"*, *"bottom"* eller *"baseline"*
Anger cellinnehållets vertikala justering.
width=*"värde"* eller *"värde%"*
Anger cellens bredd.
height=*"värde"* eller *"värde%"*
Anger cellens höjd.
colspan=*"värde"*
Anger antal kolumner cellen ska täcka.
rowspan=*"värde"*
Anger hur många rader cellen ska täcka.
bgcolor=*"#färgkod"*
Anger cellens bakgrundsfärg.
background=*"sökväg"*
Anger cellens bakgrundsbild.
bordercolor=*"#färgkod"*
Anger cellens ramfärg.
bordercolordark=*"#färgkod"*
Anger den mörka färgen i en cells 3D-ram.
bordercolorlight=*"#färgkod"*
Anger den ljusa färgen i en cells 3D-ram.
nowrap
Förhindrar radbrytningar i cellen.
<textarea>...</textarea>
Skapar ett textfält i ett formulär.
Attribut till <textarea>
cols=*"värde"*
Anger textfältets bredd mätt i antal tecken.
name=*"namn"*
Ger textfältet ett namn.
rows=*"värde"*
Anger textfältets höjd mätt i antal tecken.
wrap=*"off"*, *"virtual"* eller *"physical"*
Lägger till automatiska radbrytningar i textfältet. *Off* är grundinställningen. *Virtual* bryter orden på skärmen, men skickar texten som en rad. *Physical* skickar raden bruten.
disabled="disabled"
Anger att textfältet inte är aktiverat.
readonly="readonly"
Anger att textfältet ej är editerbart.
<tfoot>...</tfoot>
Skapar en tabellfot.
<th>...</th>
Skapar en rubrikcell.
Attribut till <th>
Samma attribut som till <td> (se denna kod)
<thead>...</thead>
Skapar ett tabellhuvud.
<title>...</title>
Skapar hemsidans titel. Placeras i <head>.
<tr>...</tr>
Skapar en rad i en tabell.

Attribut till <tr>

align=*"left"*, *"center"* eller *"right"*
Anger justering för innehållet i radens cell(er).
valign=*"top"*, *"middle"*, *"bottom"* eller *"baseline"*
Anger vertikal justering för innehållet i radens cell(er).
bgcolor=*"#färgkod"*
Anger radens bakgrundsfärg.
bordercolor=*"#färgkod"*
Anger radens ramfärg.
bordercolordark=*"#färgkod"*
Anger den mörka färgen i radens 3D-ram.
bordercolorlight=*"#färgkod"*
Anger den ljusa färgen i radens 3D-ram.

<tt>…</tt>
Gör så att texten skrivs med ett typsnitt som har fast bredd, oftast Courier.

<u>…</u>
Skapar understruken text.

…/ul>
Skapar en onumrerad lista. Varje rad i listan markeras med <li>.

<var>…</var>
Logisk variabel. Texten visas liten och med fast bredd.

<wbr />
Lägger in en mjuk radbrytning i en text som omges av <nobr>.

<xmp>…<xmp>
Logiskt textexempel. Effekt som <tt>.

Style sheets

I den här snabbreferensen hittar du attribut som kan användas i lokala, globala och länkade style sheets. Det finns två specifikationer för style sheets: CSS1 och den mer avancerade CSS2. Netscape och Microsoft stöder till stora delar CSS1, men bara delar av CSS2, och olyckligtvis inte samma delar. I denna referens har tagits med attribut som fungerar med version 4.0 av båda läsarna. Referensen är dock långtifrån komplett. Du som är intresserad hittar på bokens hemsida länkar både till W3Cs två specifikationer och Netscapes och Microsofts egna referenser.
Alla mått kan anges i punkter (pt), inch (in), centimeter (cm), millimeter (mm), pixlar (px) em eller pica (pc).

Typsnittsegenskaper

Attribut	Beskrivning	Exempel
font-size	Anger textstorlek.	{font-size: 12pt}
font-family	Anger typsnitt med antingen typsnittsnamn eller familj.	{font-family: Verdana, Helvetica, sans-serif}
font-weight	Anger typsnittets tjocklek. Kan vara normal eller bold.	{font-weight: bold}
font-style	Kan antingen vara normal eller italic, som gör texten kursiv.	{font-style: italic}

Textegenskaper

line-height	Anger avståndet till baslinjen på raden ovanför.	{line-height: 22pt}
text-decoration	Gör texten understruken (*underline*), genomstruken (*line-through*) eller normal (*none*).	{text-decoration: line-through} {color: #FF0000}
text-transform	Skapar stor begynnelsebokstav (*capitalize*), stora bokstäver (*uppercase*), små bokstäver (*lowercase*) eller inget (*none*).	{text-transform: uppercase}
text-indent	Skapar ett indrag.	{text-indent:20px}
text-align	Justerar texten. Möjliga värden: left, right, center, justify (justerad)	{text-align:right}

Blockegenskaper

margin-top	Övre marginalens storlek.	{margin-top:20px}
margin-right	Högermarginalens storlek.	{margin-right: 15mm}
margin-left	Vänstermarginalens storlek.	{margin-left: 1cm}
margin-bottom	Undre marginalens storlek.	{margin-bottom:2in}
margin	Sätter samma storlek på alla fyra marginalerna.	{margin:25px}
padding	Anger hur mycket luft det ska vara mellan blockets kant och innehåll. Kan också specificeras separat för varje sida. Se margin-bottom ovan (byt ut "margin" mot "padding").	{padding:25px}
border-width	Anger tjocklek på ram.	{border-width:25px}
border-style	Slår på (*solid*) eller av (*none)* ramen.	{border-style:solid}
Storlek och placering		
width	Anger ett blockobjekts bredd.	{width:200px}
height	Anger ett blockobjekts höjd.	{height:50px}
position	Anger varifrån positionsbestämningen ska räknas. Från fönsterkanten (*absolute*) eller från omgivande blockobjekt (*relative*).	{position:absolute}
left	Horisontell positionsbestämmelse.	{left:100px}
top	Vertikal positionsbestämmelse.	{top:100px}
z-index	Positionsbestämmelse i djupled.	{z-index:5}
float	Anger om objektet ska flyta till höger (*right*), vänster (*left*), eller inte alls (*none*).	{float:right}
Färg och bakgrund		
color	Anger en färg.	{color:#FF0000}
background-color	Anger bakgrundsfärg	{background-color:#FF0000}
background-image	Anger bakgrundsbild. Sökvägen kan vara relativ eller absolut.	{background-img:url(bilder/filt.gif)}

BILAGA 3

Alla teckenkoder

Här följer en lista över alla teckenkoder som kan användas i HTML-dokument. Instruktioner om hur teckenkoder fungerar hittar du i kapitel 18.

Tecken	Sifferkod	Teckenkod	Beskrivning
			Tab-tecken
	Radmatning
	Vagnretur
		&sp;	Mellanslag
!	!	!	Utropstecken
"	"	"	Citattecken
#	#	#	Nummertecken
$	$	$	Dollartecken
%	%	%	Procenttecken
&	&	&	Et-tecken
'	'	'	Enkelt citattecken
(	(	(	Vänsterparentes
)	)	)	Högerparentes
*	*	*	Asterisk
+	+	+	Plustecken
,	,	,	Kommatecken
-	-	‐	Minustecken
.	.	.	Punkt
:	:	:	Kolon
;	;	;	Semikolon
<	<	<	Mindre än
=	=	=	Lika med
>	>	>	Större än
?	?	?	Frågetecken
[	[	[	Vänster hakparentes
\	\	\	Backslash
]	]	]	Höger hakparentes

Tecken	Sifferkod	Teckenkod	Beskrivning
^	^	ˆ	Cirkumflex
_	_	_	Understrykning
`	`	`	Grav accent
{	{	{	Vänster vågparentes
|					Vertikal linje
}	}	}	Höger vågparentes
~	~	˜	Tilde
			Fast mellanslag
¡	¡	¡	Inv. utropstecken
¢	¢	¢	Centtecken
£	£	£	Pundtecken
	¤	¤	Valutatecken
¥	¥	¥	Yentecken
¦	¦	¦	Bruten vertikal linje
§	§	§	Paragraftecken
¨	¨	¨	Diæresis
©	©	©	Copyright
ª	ª	ª	Feminin ordinal
«	«	«	Gåsögon
¬	¬	¬	Ejtecken
–			Mjukt bindestreck
®	®	®	Registrerat varumärke
¯	¯	¯	Makron accent
°	°	°	Gradtecken
±	±	±	Plus-eller-minus
2	²	²	Superskript, 2
3	³	³	Superskript, 3

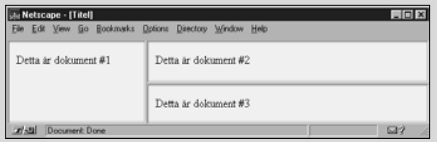
Tecken	Sifferkod	Teckenkod	Beskrivning
´	´	´	Akut accent
µ	µ	µ	Mikrotecken
¶	¶	¶	Stycketecken
·	·	·	Centrerad punkt
¸	¸	¸	Cedilj
1	¹	¹	Superscript 1
º	º	º	Maskulin ordinal
»	»	»	Gåsögon
1/4	¼	¼	En fjärdedel
1/2	½	½	En halv
3/4	¾	¾	Tre fjärdedelar
¿	¿	¿	Inv. frågetecken
À	À	À	Stora A grav
Á	Á	Á	Stora A akut
Â	Â	Â	Stora A cirkumflex
Ã	Ã	Ã	Stora A tilde
Ä	Ä	Ä	Stora Ä
Å	Å	Å	Stora Å
Æ	Æ	Æ	Ligatur
Ç	Ç	Ç	Stora C, cedilj
È	È	È	Stora E grav
É	É	É	Stora E akut
Ê	Ê	Ê	Stora E cirkumflex
Ë	Ë	Ë	Stora E diæresis
Ì	Ì	Ì	Stora I grav
Í	Í	Í	Stora I akut
Î	Î	Î	Stora I cirkumflex
Ï	Ï	Ï	Stora I diæresis
Ð	Ð	Ð	Stora Eth
Ñ	Ñ	Ñ	Stora N tilde
Ò	Ò	Ò	Stora O grav
Ó	Ó	Ó	Stora O akut
Ô	Ô	Ô	Stora O cirkumflex
Õ	Õ	Õ	Stora O tilde
Ö	Ö	Ö	Stora Ö
x	×	×	Gångertecken
Ø	Ø	Ø	Danskt Ö
Ù	Ù	Ù	Stora U grav

Tecken	Sifferkod	Teckenkod	Beskrivning
Ú	Ú	Ú	Stora U akut
Û	Û	Û	Stora U cirkumflex
Ü	Ü	Ü	Tyskt Y
Ý	Ý	Ý	Stora Y akut
Þ	Þ	Þ	Stora Thorn
ß	ß	ß	Dubbel-S
à	à	à	Lilla a grav
á	á	á	Lilla a akut
â	â	â	Lilla a cirkumflex
ã	ã	ã	Lilla a tilde
ä	ä	ä	Lilla ä
å	å	å	Lilla å
æ	æ	æ	Lilla danska ä
ç	ç	ç	Lilla c cedilj
è	è	è	Lilla e grav
é	é	é	Lilla e akut
ê	ê	ê	Lilla e cirkumflex
ë	ë	ë	Lilla e diæresis
ì	ì	ì	Lilla i grav
í	í	í	Lilla i akut
î	î	î	Lilla i cirkumflex
ï	ï	ï	Lilla i diæresis
ó	ð	ð	Lilla eth
ñ	ñ	ñ	Lilla n tilde
ò	ò	ò	Lilla o grav
ó	ó	ó	Lilla o akut
ô	ô	ô	Lilla o cirkumflex
õ	õ	õ	Lilla o tilde
ö	ö	ö	Lilla ö
÷	÷	÷	Divisionstecken
ø	ø	ø	Litet danskt ö
ù	ù	ù	Litet u grav
ú	ú	ú	Litet u akut
û	û	û	Litet u cirkumflex
ü	ü	ü	Litet tyskt y
ý	ý	ý	Litet y akut
þ	þ	þ	Lilla thorn
ÿ	ÿ	ÿ	Lilla y diæresis

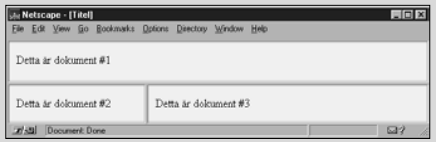
BILAGA 4

Lathund för rutsystem

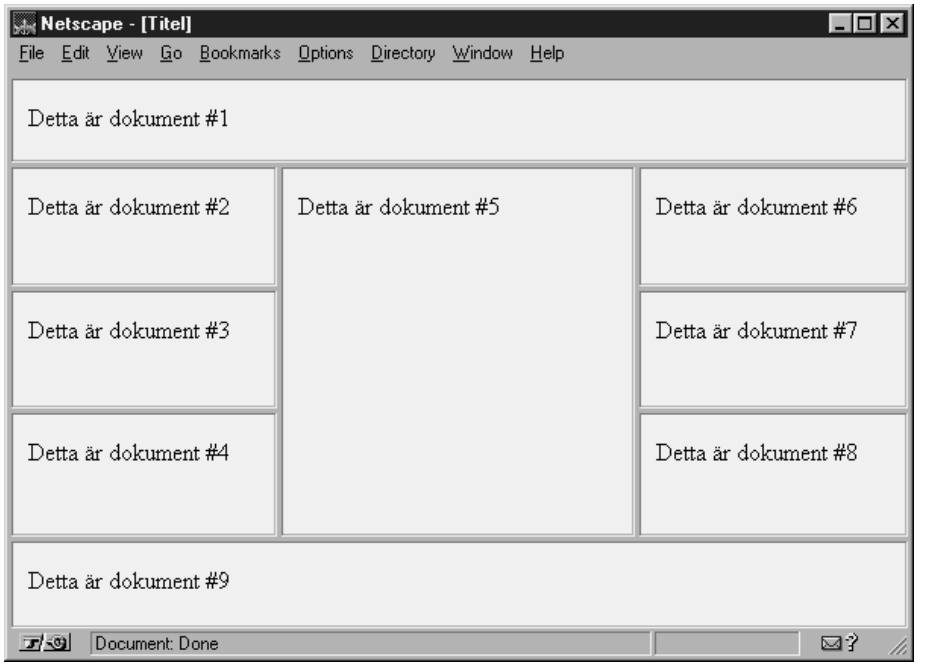
Att kombinera rader och kolumner i ett rutsystem kan vara knepigt. I den här lathunden hittar du fyra system som du kan modifiera så att de passar din hemsida.



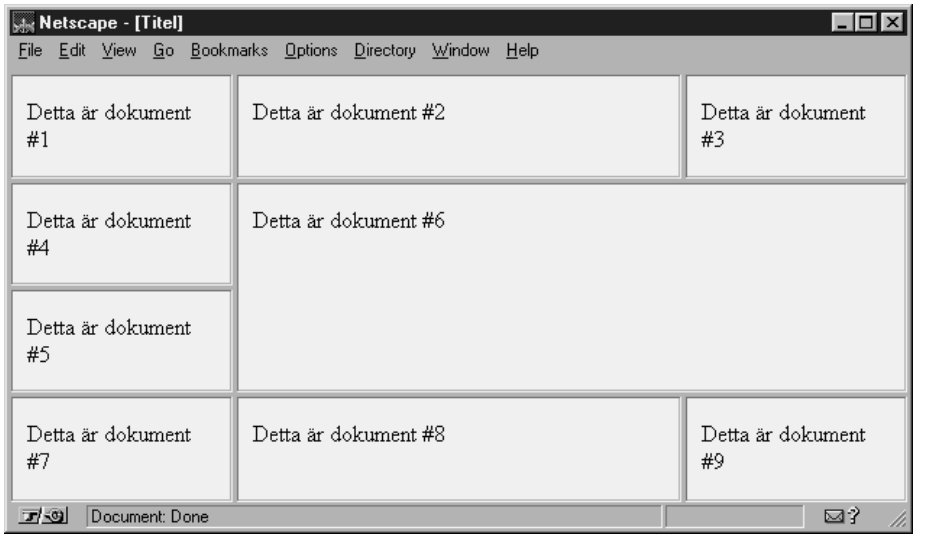

```
<html>

<head><title>Titel</title></head>

<frameset cols="200,*">

<! En ruta som hamnar i första kolumnen>
<frame src="dok1.html" />

<! Ett nytt rutsystem som hamnar i andra kolumnen>
<frameset rows="60,*">

<! En ruta som hamnar på första raden>
<frame src="dok2.html" />

<! En ruta som hamnar på andra raden>
<frame src="dok3.html" />

</frameset>

</frameset>

</html>
```




```
<html>

<head><title>Titel</title></head>

<frameset rows="60,*">

<! En ruta som hamnar på första raden>
 <frame src="dok1.html" />

<! Ett nytt rutsystem som hamnar på andra raden>
        <frameset cols="200,*">

        <! En ruta som hamnar i första kolumnen>
        <frame src="dok2.html" />

        <! En ruta som hamnar i andra kolumnen>
        <frame src="dok3.html" />

        </frameset>

</frameset>

</html>
```




```
<html>
<head><title>Titel</title></head>

<frameset rows="60,*,60">

<frame src="dok1.html" />
        <frameset cols="180,*,180">

        <frameset rows="*,*,*">
                <frame src="dok2.html" />
                <frame src="dok3.html" />
                <frame src="dok4.html" />
        </frameset>
<frame src="dok5.html" />

        <frameset rows="*,*,*">
                <frame src="dok6.html" />
                <frame src="dok7.html" />
                <frame src="dok8.html" />
        </frameset>
        </frameset>
<frame src="dok9.html" />
</frameset>
</html>
```


```
<html>
<head><title>Titel</title></head>

<frameset rows="25%,50%,25%">

        <frameset cols="25%,*,25%">
                <frame src="dok1.html" />
                <frame src="dok2.html" />
                <frame src="dok3.html" />
        </frameset>

        <frameset cols="25%,*">

                <frameset rows="25%,25%">
                        <frame src="dok4.html" />
                        <frame src="dok5.html" />
                </frameset>
                <frame src="dok6.html>
        </frameset>

        <frameset cols="25%,*,25%">
                <frame src="dok7.html" />
                <frame src="dok8.html" />
                <frame src="dok9.html" />
        </frameset>

</frameset>

</html>
```

Sakregister

A

<!> kod, 34, 35
<a> kod, 37, 48
absoluta sökvägar, se sökvägar
ankare, 44
<area> kod, 79, 73
ASCII, 187
asterisk, 116
attribut, 26

B

<b> kod, 17, 35
bakgrundsbild, se bilder
bakgrundsfärg, se färg
bakgrundsljud, 174
<basefont> kod, 29, 35
<bgsound> kod, 174
bilder, 55
flytande, 61
som länkar, 63
storlek, 66
alternativtext, 67
som bakgrund, 68, 104
transparenta, 69
i tabeller, 102, 103
bildformat, 56
bildkartor, 71
serverbaserade, 73
klientbaserade, 79
falska, 82
<blink> kod, 24, 35
<blockquote> kod, 32, 35
<body> kod, 19, 35
**
 kod, 25, 35, 70**
bugg i Netscape, 59, 64

C

cache-minne, 16
<caption> kod, 92, 110
** kod, 32, 35**
CERN, 3
CGI, 139
cgi-bin 140
CGIWrap, 140
<col> kod, 175, 180
<colgroup> kod, 176, 180
Common Gateway Interface, 139
Courier, 23

D

<dd> kod, 40, 42
** kod, 32, 35, 60,153**
<dl> kod, 40, 42
<dt> kod, 40, 42

E

enkäter, 127
e-post, 53

F

fast mellanslag, 190
felsökning, 34
FETCH program, 135
File Transfer Protocol, 43
filnamn, 13
förlängning, 56
flytande rutor, se <iframe>
<font> kod, 28, 35
<form> kod, 128, 134
formulärobjekt, 130
<frame /> kod, 114, 125
frames, se rutsystem
<frameset> kod, 113, 125
FTP, 43, 135
-program, 135
färg, 85
i tabeller, 102, 175
som bakgrund, 85
färgkoder, se färg
förlängning, se filnamn

G

Gif-bilder, 56
GIF89a, 69

H

<head> kod, 19, 35
hemkatalog, 136
hexadecimala talsystemet, 87
<hr /> kod, 26, 35
HTML 5, 17
<html> kod, 19, 35
HTML-dokument
namn, 13
struktur, 18
skrivskydda, 21
<hvärde> kod, 24, 35
hyperlänkar, 3, 37
interna, 44
i nytt fönster, 47
hypertext, 3
HyperText Markup Language, se HTML
hårddiskminne, 8

I

<i> kod, 22, 35
<iframe> kod, 175, 179
<img /> kod, 56, 70, 83, 183

<input /> kod, 130, 134
INRIA, 9
interna länkar, se hyperlänkar
Internet Explorer, 10, 171
internminne, 8

J
Java, 160
JavaScript, 160
Jpeg-bilder, 57

K
kartfil, 75, 84
CERN-modell, 75
NCSA-modell, 76

L
** kod, 50, 54**
linjer, se <hr />
<link /> kod, 155, 157
listor, 37
onumrerade, 38
numrerade, 39
länkadress, 44
länkar, se hyperlänkar

M
malldokument, 20
<map> kod, 79, 73
marginal
med <dl>, 41
<marquee> kod, 172, 179
Massachusetts Institute of Technology, se MIT
Microsoft, 10
minne, 8
MIT, 9
mjukt bindestreck, 190
Mosaic, 9
<multicol> kod, 149, 152

N
NCSA, 5
Netscape Navigator, 9, 181
<nobr> kod, 33, 35
<noframes> kod, 124, 126

O
** kod, 39, 42**
<option> kod, 133, 134
ordbehandlingsprogram, 8

P
<p> kod, 25, 35
<pre> kod, 27, 35
public_html, 137

R
radavstånd, 144
ramar
på bilder, 65
i tabeller, 100, 175
i rutsystem, 122
relativa sökvägar, se sökvägar
rubriker, 24
i tabeller, 104
rullande text, se <marquee>
rutor, se rutsystem
rutsystem, 111
kolumner, 113
rader, 115
länkar i, 120
rättigheter, 138

S
<s> kod, 23
<script> kod, 161
<SELECT> kod, 133, 134
shareware, 189
Skriv text, program, 8
skrivskydd, 21
<spacer> kod, 182, 186
<style> kod, 145, 158
style sheets, 141
lokala, 143
globala, 144
länkade, 155
<sub> kod, 24
<sup> kod, 24
sökmotor, 19
sökvägar, 46
relativa, 46, 48
absoluta, 49

T
tabeller, 89
rader, 91
kolumner, 91
bredd och höjd, 97
<table> kod, 90, 108
<tbody> kod, 175, 180
<td> kod, 91, 109
teckenkoder, 145
<textarea> kod, 133, 134
texteditorer, 7
<tfoot> kod, 175, 180
<th> kod, 91, 109
<thead> kod, 175, 180
<title>, kod, 19, 35
<tr> kod, 90, 109
<tt> kod, 22, 35
typografi, 144
typsnitt, 31, 144

U
<u> kod, 23, 35
** kod, 50, 54**
upphovsrätt, 58
uppladdning, 135
upplösning, 96

V/W
versioner av program, 9
W3C, 9
webläsare, 9
webserver, 5, 135
World Wide Web, 4
World Wide Web Consortium, se W3C
WS_FTP program, 135