Hur man gör webbsidor

Instruktioner, koder, exempel och kommentarer

HTML 4.0
Cascading Style Sheets
Common Gateway Interface
Server-side includes
Magic Cookies
Ljud, film och multimedia

Av Anders Hultman
8:e upplagan

Hur man gör webbsidor

8:e upplagan, februari 1998

Tryckort: Stockholm

Innehåll

Förord

Den här boken handlar om hur man gör webbsidor. Den riktar sig till den som vill publicera information på Internet eller ett intranät, för sitt företags, avdelnings, förenings eller organisations räkning eller av privat intresse.

Boken går igenom version 4.0 av uppmärkningsspråket HTML samt designprinciper och god smak för publicering på www. Boken tar även upp cascading style sheets, common gateway interface, server-side includes, magic cookies, ljud, film och multimedia.

Boken tar inte upp betalsystem på www eller avancerade system för hel- eller halvautomatisk publicering och löpande uppdatering av stora mängder information, som till exempel order- och lagerdatabaser eller sådana system som används av tidningsredaktioner.

För att förstå denna bok bör du ha viss datavana, som att behärska de grundläggande funktionerna i en persondator, att känna till begrepp som fil/dokument, mapp/directory och hårddisk, samt att du har använt Internet och world wide web för att titta på andra företags och personers webbsidor.

De viktigaste och vanligaste funktionerna, som text med olika utseende och funktion, länkar och bilder, listor och tabeller, gås igenom i bokens första hälft. Den andra hälften handlar om mer avancerade finesser som frames, imagemaps, formulär och dynamiska webbsidor. Sista avsnittet innehåller en lista över adresser till webbsidor med mer information för den som vill fördjupa sig inom något område som boken tar upp. Det går bra att enbart läsa de första avsnitten och därigenom lära sig de viktigaste funktionerna först, för att bygga på med fler och fler finesser vid ett senare tillfälle. Det går också bra att använda boken som en uppslagsbok eller referensverk.

Varken denna bok eller www i sig är bunden till någon speciell programvara eller datortyp, men de exempel som tas upp är gjorda med webbläsaren Netscape Navigator under Windows 95 och servrarna NCSA och Apache under Unix.

Hur man gör webbsidor

En webbsida är ett HTML-dokument tolkat och presenterat av en webbläsare. Ett HTML-dokument är en textfil som innehåller vissa styrkoder, s.k. *tags* eller *elements*. Med styrkoderna markerar man rubriker, vilken text som ska vara kursiv eller fet, bilders placering etc. Styrkoderna anger också att viss text ska fungera som länkar till andra webbsidor, s.k. hypertext. HTML betyder ”hypertext markup language”.

Styrkoderna tolkas av en webbläsare, som är det program som du använder för att titta på webbsidor med. De två vanligaste webbläsarna heter Netscape Communicator och Microsoft Internet Explorer, men det finns många andra, t.ex. Opera, Arena, Mosaic och Lynx.

När du klickar fram en ny sida hämtar webbläsaren ett HTML-dokument, tolkar styrkoderna, lägger ut fönstret efter dem och efter dina personliga inställningar.

För att skriva HTML-dokument kan du använda en mängd olika program. Det finns tre grupper av program med något olika arbetssätt, och vilken sorts program du ska välja beror dels på hur avancerade sidor du vill kunna göra, dels på hur mycket du vill ha kontroll över själv och hur mycket du vill – och vågar – överlåta åt programmet.

Texteditorer

Den första gruppen av program för att skriva HTML-dokument är vanliga texteditorer, till exempel Anteckningar, Notepad och Wordpad för Windows eller Skriv Text för Macintosh. Editorer som BBEdit och Emacs hör också till denna grupp. Det går också bra att använda en ordbehandlare som Microsoft Word, men då måste man komma ihåg att spara som ”Text Only” eller motsvarande.

Använder du något av dessa program får du komma ihåg alla styrkoder i huvudet och skriva in dem för hand. Du ser inte hur sidan kommer att se ut, utan måste ha ett webbläsarfönster öppet bredvid texteditorns fönster. Efter varje gång du gjort en ändring i koden trycker du på ”Reload”, ”Uppdatera” eller motsvarande knapp i webbläsaren för att se de ändringar du gjort.

HTML-editorer

Nästa kategori av program är s.k. HTML-editorer. Precis som med de enklaste texteditorerna så ser du den rena HTML-koden i editorns fönster, och har ett webbläsarfönster (t.ex. Netscape) öppet bredvid. Skillnaden är den att programmet har knappar, menyval och andra funktioner som gör att du slipper skriva in alla koder manuellt. Vissa av dessa visar HTML-koderna i en avvikande färg, men principen är densamma. Exempel på program i denna grupp är Homesite, HotDog och PageSpinner. Adresser till webbsidor där man kan hämta dessa program finns i kapitlet ”Mer information”.

WYSIWYG-editorer

Den tredje gruppen program är verktyg där du jobbar med sidan utan att se koden alls, s.k. WYSIWYG-editorer (what you see is what you get). De låter dig se hur webbsidan ser ut medan du skriver den. Du behöver inte lära dig några HTML-koder alls, utan drar och släpper text och bilder ungefär som i ett layoutprogram. Exempel på program i denna grupp är FrontPage, PageMill, JoyHTML och Netscape Composer. Dessa program brukar vara mycket populära bland nybörjare, och man kommer snabbt igång med att göra webbsidor. Men de flesta av dessa program har också begränsningar. Till exempel brukar de inte stödja alla HTML-koder som finns, och den kod som dessa program genererar brukar vara fylld med onödiga kommandon och klumpiga lösningar för att tvinga sidan till en viss layout. Dessutom kan man inte vara säker på att det man ser verkligen är det man får (se även avsnittet ”Lita på dina klienter”).

Flera av dessa program säljs i datorbutiker, andra går att hämta på Internet. Adresser till webbsidor där man kan hämta dem finns i kapitlet ”Mer information”.

Förutom olika typer av editorer finns det även konverteringsprogram som gör om t.ex. ordbehandlarfiler av olika typer till HTML. För att få ett bra resultat med dessa så måste ordbehandlarfilerna vara mycket väl strukturerade (se även avsnittet ”Lita på dina klienter”).

Styrkoder

Styrkoderna i HTML ringas in med mindre än- och större än-tecken. Vissa koder jobbar i par (startkod och slutkod) medan andra förekommer ensamma. Exempel:

<KOD> (startkod eller ensam kod)
</KOD> (slutkod)

Till många av koderna kan man lägga attribut, d.v.s. kommandon som mer i detalj påverkar vad koden gör:

<KOD SIZE="3" BORDER="2" ALIGN="CENTER" NORESIZE>

De flesta attributen är i formen av namn/värde-par, som size, border och align i exemplet ovan. Namn och värde separeras med ett likamed-tecken. Värdet (3, 2 respektive center i exemplet ovan) sätts inom citattecken. I vissa fall kan citattecknen utelämnas, men det är aldrig fel att ha dem med. Vissa attribut behöver inte något värde (som noresize i exemplet ovan). Har man flera attribut till samma kod skriver man dem efter varandra, separerade med mellanslag. Den inbördes ordningen spelar ingen roll.

Exempel: om man vill ha en rubriktext använder man styrkoden <H1> (header level 1). Man ringar då in rubriktexten med startkod och slutkod:

<H1>Rubrik</H1>

Rubriken blir normalt vänsterställd. Om man vill att rubriken ska bli centrerad kan man ange det som ett attribut till rubrik-koden: <H1 ALIGN="CENTER">. Rubriken avslutas med </H1> som vanligt.

Det spelar ingen roll om du skriver styrkoderna med små eller stora bokstäver. Man kan till och med blanda inom en och samma kod.
,
,
 och
 ger precis samma resultat, en fast radbrytning. Vissa anser att man alltid ska använda stora bokstäver i koderna, detta för tydlighetens skull när man redigerar filen i efterhand, andra skriver konstant med små bokstäver för att slippa trycka på skift hela tiden. I denna handledning har jag konsekvent skrivit styrkoderna med stora bokstäver.

HTML-specifikationen utvecklas hela tiden, och nya koder kommer till. Olika webbläsarfabrikat hittar på egna tillägg till HTML som ibland blir upptagna i specifikationen senare, tillägg som nya, men inte äldre, versioner av webbläsarna stöder. Om man har skrivit en styrkod som en viss webbläsare inte ”känner till” så kommer den bara att hoppa över den utan åtgärd. Till exempel så kan inte Netscape version 1.0 byta färg på texten. Den webbläsaren kommer att hoppa över färgbytarkoden och visa texten svart som vanligt. På samma sätt kommer alla webbläsare att hoppa över felstavade styrkoder.

Lita på dina klienter

Exakt hur texten kommer att se ut på skärmen kan du aldrig veta när det gäller koder som t.ex. <H1>. Det beror på att det ställs in i varje enskild webbläsare. Netscape i grundutförande visar <H1> som 24 punkters halvfet Times, men det är bara Netscapes egen tolkning, det finns inget i HTML som säger att det ska vara så. Andra webbläsare kan välja att visa olika rubriknivåer på ett helt annat sätt.

Utseendet skiljer sig inte bara mellan olika webbläsarprogram, utan även mellan olika datorer som kör samma programversion. De flesta webbläsare tillåter till exempel att användaren själv väljer typsnitt, storlek på brödtexten och sidornas bakgrundsfärg. Andra saker såsom skärmtyp, fönsterstorlek, färgmöjligheter etc kan också inverka på hur sidan ser ut hos användaren.

För den som är van vid att producera saker som trycks på papper kan detta verka lite konstigt. Det beror på att HTML *inte* är ett språk som beskriver hur en sida *ser ut*, utan ett märkningsspråk (markup language) som beskriver *innehåll och struktur*. HTML-koderna markerar nämligen textens *funktion* (till exempel ”Rubrik på översta nivån”) och inte textens *utseende* (till exempel ”24 punkter halvfet Times”). Utseendet bestäms av webbläsarprogrammet och användarens personliga inställningar.

Anledningen är att webben ska vara helt plattformsoberoende. Alla har inte samma typsnitt installerade, alla datorskärmar kan inte visa text i olika storlekar, alla datorer har inte ens skärm. Det finns webbläsare som visar texten på en punktskriftsdisplay, och man kan tänka sig webbläsare som läser upp texten med talsyntes, till exempel för användning av synskadade eller vid bilkörning.

Begreppet WYSIWYG (what you see is what you get) är egentligen inte tillämpligt på webben. Tänkesättet skiljer sig alltså från ordbehandlare och layoutprogram där man har full kontroll över sidornas exakta utseende. Därför är det svårt att konvertera ordbehandlingsdokument direkt till HTML, för det är inte självklart hur till exempel text i olika storlekar ska tolkas; är det en rubrik, eller är det bara större text?

Det kan känna lite osäkert att överlåta till läsaren hur din sida ska se ut, men var inte orolig för det. Du måste lita på dina ”klienter”. Låt dem själva avgöra hur de vill ha det på sina skärmar, du har ändå inget val.

HTML-filens struktur

Börja HTML-filen med att tala om att det är just en HTML-fil. På första raden i HTML-filen ska det stå en rad som anger vilken version av HTML-specifikationen som dokumentet följer. Formellt sett så är denna rad obligatorisk, men det är många som utelämnar den. Exempel:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">

Denna deklaration anger att dokumentet följer version 4.0 av HTML-specifikationen.

Efter dokumenttypsdeklarationen markerar man att själva HTML-koden börjar med hjälp av styrkoden **<HTML>**. Avsluta filen med motsvarande **</HTML>**. Denna styrkod är inte obligatorisk, men bör användas för tydlighetens skull.

Head och body

En HTML-fil delas in i två stora delar; ”head” och ”body”. I head-delen läggs saker som *inte* ska vara med på själva sidan, och i body-delen läggs saker som *ska* vara med på själva sidan. Styrkoderna ser ut så här:

<HEAD> </HEAD> respektive
<BODY> </BODY>

Dessa styrkoder är inte obligatoriska, men bör användas för tydlighetens skull.

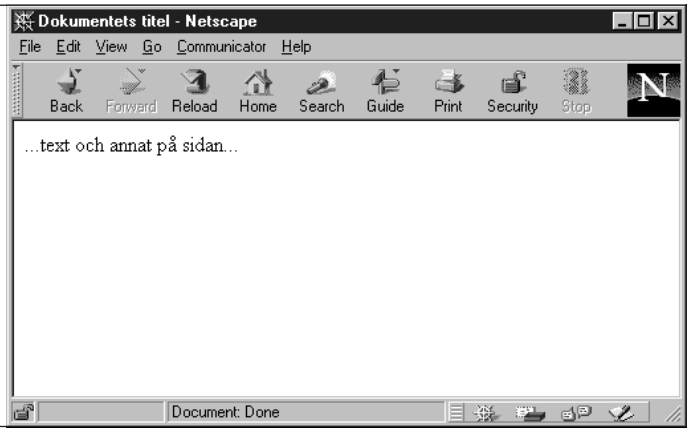
Dokumentets titel

Dokumentets titel är den text som bland annat står i webbläsarfönstrets namnlist. Titeln markeras med styrkoderna **<TITLE>** och **</TITLE>**.

Titeln är inte med på själva sidan, och ska alltså ligga mellan <HEAD> och </HEAD>. Titeln ska vara en slags överskrift till sidan. Det är viktigt att ange en meningsfull titel, eftersom det är den som kommer synas i bokmärkeslistor och sökmaskiner. Titeln får vara maximalt 64 tecken, inklusive mellanslag.

En sidas uppbyggnad ser i stort ut enligt följande:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Dokumentets titel</TITLE>
...eventuella andra head-element...
</HEAD>
<BODY>
...text och annat på sidan...
</BODY>
</HTML>
```



Netscape Navigator 4.0 visar sidan i exemplet ovan på detta sätt.

Se hur andra har gjort

Vill du så kan du ”tjuvkika” på hur andra personers webbsidor är uppbyggda. I de flesta webbläsare så finns det ett menyval eller liknande som gör så att den rena HTML-koden syns. I Netscape heter denna funktion ”View Document Source” (version 2 och 3) eller ”View Page Source” (version 4). Denna funktion är mycket användbar om man vill lära sig strukturen i HTML genom andras exempel.

Stycken, radbrytningar och block

För att skriva vanlig löpande text behöver du inte ange några styrkoder. Det är bara att prata på. Webbläsaren bryter raden automatiskt om fönstret inte räcker till längre. Det gör ingenting om du väljer att lägga in några extra radbrytningar i HTML-filen. Webbläsaren struntar i dem. Detsamma gäller mellanslag. Har du lagt in flera mellanslag i rad så lägger webbläsaren i alla fall bara ut ett.

Nytt stycke

För att dela in texten i stycken använder man styrkoden **<P>** (står för paragraph, d.v.s. engelska för stycke). <P>-koden kan avslutas med **</P>** om man vill, men det är inte nödvändigt.

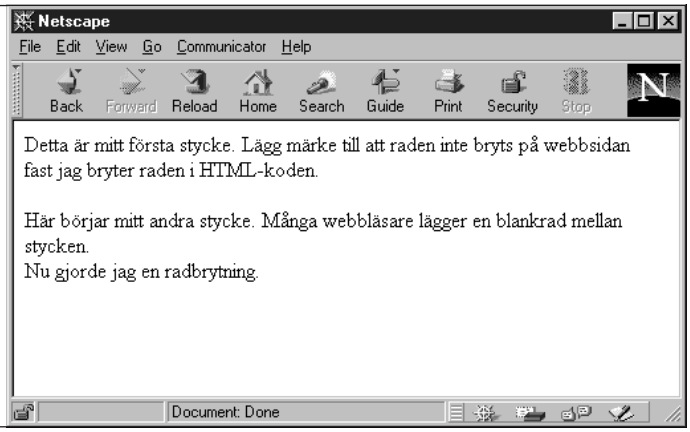
Netscape och många andra webbläsare lägger normalt en blankrad mellan två stycken. Man kan inte få större avstånd mellan två stycken genom att upprepa <P>-koden flera gånger, eftersom <P> betyder nytt stycke, inte blankrad. Vill man ha större eller mindre avstånd kan man använda style sheets (se detta avsnitt).

Dela in din text i lagom långa stycken. En text utan styckesindelningar är hopplös att läsa, textmassan blir oöverskådlig.

Ny rad

Om man vill tvinga texten att börja på ny rad använder man **
** (break) som ger en hård radbrytning. Den här styrkoden klarar sig själv och ska inte ha någon slutkod.

```
Detta är mitt första stycke.
Lägg märke till att raden inte
bryts på webbsidan fast jag
bryter raden i HTML-koden.
<P>
Här börjar mitt andra stycke.
Många webbläsare lägger en
blankrad mellan stycken.<BR>
Nu gjorde jag en radbrytning.
```



Hårt mellanslag

Om man vill hindra att raden bryts mellan två ord använder man koden ** ** istället för vanligt mellanslag. Koden ger ett ”non breaking space”, ett fast mellanslag. Observera att   måste skrivas med små bokstäver. Det går inte att skriva

Mjukt bindestreck

Om man vill tillåta att ett långt ord avstavas i en viss punkt kan man använda koden **** som ger ett mjukt bindestreck (soft hyphen). Observera att måste skrivas med små bokstäver. Det går inte att skriva &SHY; Använd denna kod med försiktighet eftersom äldre webbläsare kanske visar koden på själva sidan. Prova dig fram.

Hindra radbrytning

Om du har ett textavsnitt som webbläsaren inte får radbryta, till exempel ett telefonnummer där de olika siffergrupperna i numret annars skulle kunna hamna på olika rader, kan du använda styrkoden **<NOBR>** (no break). Glöm inte bort att avsluta med **</NOBR>**, annars får du en enda, väldigt lång rad. Denna kod är Netscapes eget påhitt och stöds inte av andra webbläsare.

Förformaterad text

Text inom styrkoderna **<PRE>** och **</PRE>** visas av webbläsaren likadant som den står i HTML-filen. Texten visas med ett typsnitt med fast breddsteg (”skrivmaskinstext”) och alla mellanslag och radbrytningar blir som i filen. Detta är bra för till exempel programlistningar eller när man har en ren textfil som man inte vill göra om till HTML-kod.

Det går att ange en maximal bredd för PRE-fältet. Bredden anges i antal tecken, till exempel <PRE WIDTH="80">. Detta attribut stöds inte av alla webbläsare.

Inom ett block med PRE-markerad text kan det inte finnas styrkoder som ändrar textens storlek eller utseende. Vissa koder, som till exempel länkar och ankare, går dock bra.

Gruppera element

För att gruppera ihop olika element till block används **<DIV>**. Man kan blanda t.ex. stycken, rubriker, listor, tabeller etc och gruppera ihop dem till ett block. Blocket avslutas med **</DIV>**. Man kan sedan ange stil för all text i hela blocket med hjälp av style sheets (se detta avsnitt). Man kan också vänsterställa, centrera eller högerställa blocket med <DIV **ALIGN="LEFT"**>, <DIV **ALIGN="CENTER"**> respektive <DIV **ALIGN="RIGHT"**>.

Ett block inneslutet mellan <DIV> och </DIV> formar ett eget stycke. Det blir normalt blankrader av samma typ som mellan stycken före och efter blocket. För att gruppera objekt, till exempel vissa ord, inom ett och samma stycke eller inom en och samma rad använder man **<SPAN>** som avslutas med **</SPAN>**.

Varken <DIV> eller <SPAN> ger texten någon speciell stil av sig själva, utan stilen åstadkoms via style sheets (se detta avsnitt).

Rubriker

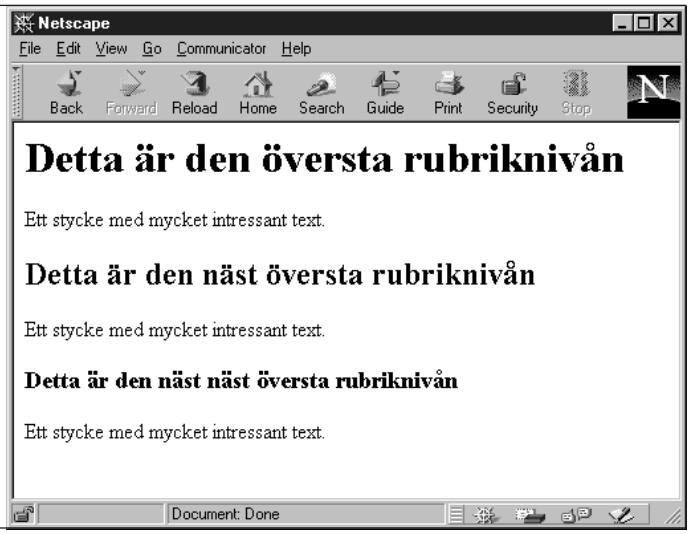
Rubriker markeras med styrkoden **<Hx>** där x är en siffra mellan 1 och 6. <H1> är den viktigaste rubriken, d.v.s. den högsta rubriknivån, <H2>, den näst viktigaste o.s.v. medan <H6> är den lägsta. <H1> brukar därför visas på skärmen med störst stil och <H6> med minst.

Dessa styrkoder används också av många för att ändra storleken på vanlig text, vilket egentligen inte är så lyckat. Använd style sheets istället (se detta avsnitt).

En <Hx>-kod måste alltid avslutas med en matchande </Hx>, d.v.s. <H1> avslutas med </H1> och <H2> avslutas med </H2> o.s.v. Netscape lägger normalt en blankrad före och efter en rubrik skapad med <Hx>.

HTML-specifikationen rekommenderar att man inte ”hoppar över” rubriknivåer. <H1> </H1> ska alltså följas av <H2> </H2> som följs av <H3> </H3> o.s.v. Tycker man att rubrikernas storlek på skärmen inte passar ens syften kan man förändra utseendet på dem med hjälp av sheets istället (se detta avsnitt).

```
<H1>Detta är den översta
rubriknivån</H1>
Ett stycke med mycket
intressant text.
<H2>Detta är den näst
översta rubriknivån</H2>
Ett stycke med mycket
intressant text.
<H3>Detta är den näst näst
översta rubriknivån</H3>
Ett stycke med mycket
intressant text.
```



Strukturerad text

Det finns många styrkoder för att ändra stilen på texten, och principen bakom de flesta är att de markerar vad textens syfte är, till exempel att texten ska betonas, att den är ett citat eller en definition. Det finns också styrkoder som mer direkt styr textens utseende, koder som till exempel ger fet eller kursiv stil.

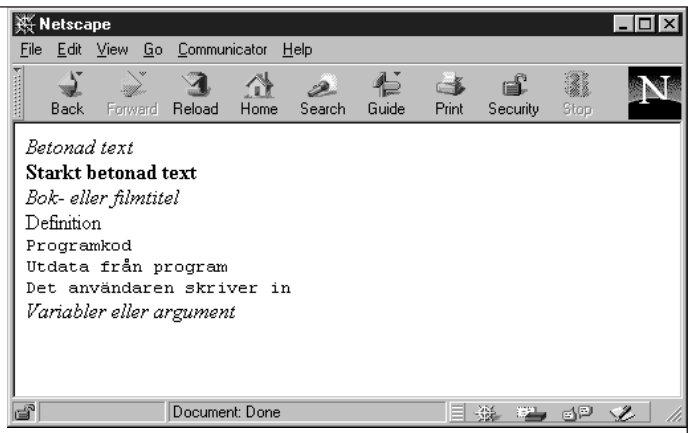
Text markerad med någon av dessa koder kan lätt fås att visas precis på önskat sätt med hjälp av style sheets (se detta avsnitt).

Fraselement

Med fraselement markerar man korta bitar av texten, ibland bara ett enda ord, och anger därmed textbitens strukturella funktion i texten. Varje markeringstyp och varje markerad textbit kan ges ett visst utseende med hjälp av style sheets (se detta avsnitt).

<EM> </EM> Emphasis. Betonad, eftertrycklig text. Visas ofta med *kursiv stil*.
<STRONG> </STRONG> Starkt betonad text. Visas ofta med **fet stil**.
<CITE> </CITE> Titlar på böcker, filmer, etc. Visas ofta med *kursiv stil*.
<DFN> </DFN> Definition av en term. Visas ofta med *kursiv stil*.
<CODE> </CODE> Programkod. Visas ofta med fast bredd, t.ex. `Courier`.
<SAMP> </SAMP> Återgivning av utdata från ett datorprogram. Visas ofta med fast bredd, t.ex. `Courier`.
<KBD> </KBD> User keyboard entry. När du talar om vad en användare ska skriva in. Bör visas som fet text med fast bredd, t.ex. **`Courier`**, men många webbläsare visar den med vanlig, mager `Courier`.
<VAR> </VAR> För variabler eller argument till ett datorprogram. Visas ofta med *kursiv stil*.
<ACRONYM> </ACRONYM> För förkortningar. Denna kod ger inte texten någon speciell stil (se nedan).

```
<EM>Betonad text</EM><BR>
<STRONG>Starkt betonad
text</STRONG><BR>
<CITE>Bok- eller
filmtitel</CITE><BR>
<DFN>Definition</DFN><BR>
<CODE>Programkod</CODE><BR>
<SAMP>Utdata från
program</SAMP><BR>
<KBD>Det användaren
skriver in</KBD><BR>
<VAR>Variabler eller
argument</VAR><BR>
```



Så här visar Netscape 4.0 olika fraselement.

Koderna <EM> och <STRONG> är användbara för att betona text i allmänna sammanhang, medan de andra främst är meningsfulla i teknisk dokumentation.

Koden <ACRONYM> låter författaren till texten tydligt visa vilka ord som ska tolkas som förkortningar, vilket är bra för stavningskontrollprogram och talsyntesbaserade webbläsare. Till <ACRONYM> kan man lägga attribut som visar vad förkortningen står för, och på vilket språk den är. Exempel:

```
<ACRONYM TITLE="World Wide Web">WWW</ACRONYM>
<ACRONYM TITLE="Europeiska Unionen" LANG="sv">EU</ACRONYM>
```

Citat

När man vill lägga in längre stycken som separeras från resten av texten (t.ex. citat) kan man använda styrkoderna **<BLOCKQUOTE>** och **<Q>**. De avslutas med **</BLOCKQUOTE>** respektive **</Q>**.

<BLOCKQUOTE> används när citatet ska utgöra ett helt, eget stycke. Många webbläsare gör ett indrag för att skilja citatet från resten av texten på sidan. En del personer använder <BLOCKQUOTE> som ett allmänt sätt att åstadkomma indrag, vilket är vanskligt eftersom andra webbläsare kan välja att särskilja citat på helt andra sätt, till exempel med >-tecken enligt praxis för citat i e-post och Usenet News. Använd style sheets istället (se detta avsnitt).

<Q> används när citatet är en del av den löpande texten. Det blir då inte indraget. Styrkoden <Q> accepteras inte ännu av alla webbläsare.

I både <BLOCKQUOTE> och <Q> kan man använda attributet CITE="url" för att ange en webbadress varifrån citatet är lånat.

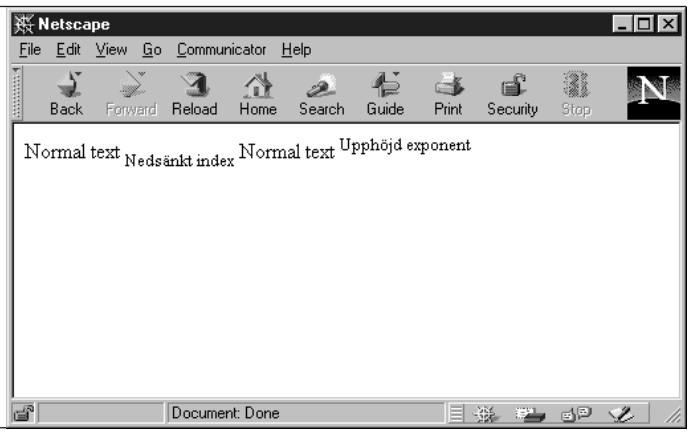
Utseendet på båda typerna av citat, såväl som individuella citat, kan bestämmas med hjälp av style sheets (se detta avsnitt).

Exponent och index

Exponent och index används i vissa tekniska sammanhang, t.ex. m^2 och H_2O, samt som förkortningar i vissa språk, t.ex. franskans ”M^{lle} Dupont”

**** Visar texten som en upphöjd exponent (t.ex. m^2).
**** Visar texten som ett nedsänkt index (t.ex. H_2O).

```
Normal text
<SUB>Nedsänkt index</SUB>
Normal text
<SUP>Upphöjd exponent</SUP>
```



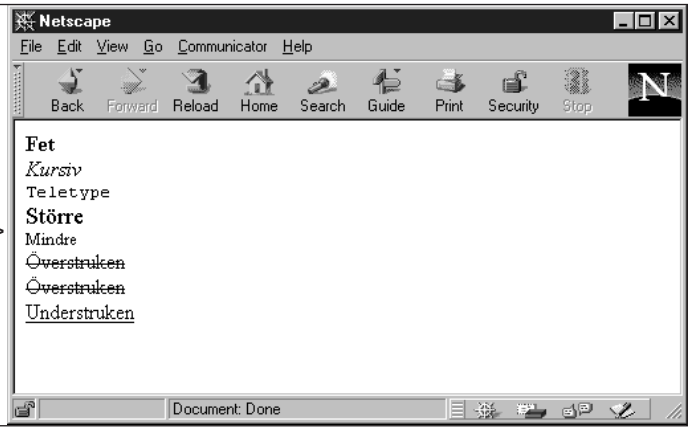
Teckenstilelement

Med teckenstilelement markerar man korta bitar av texten, ibland bara ett enda ord, och markerar därmed textbitens stiltyp, till exempel fet eller kursiv. Att använda dessa styrkoder uppmuntras inte i HTML-specifikationen, eftersom de bara är meningsfulla i grafiska webbläsare.

Det exakta utseendet av text markerad med dessa koder beror på webbläsaren, och kan även påverkas med style sheets (se detta avsnitt).

<B> </B> Bold. **Fet stil**.
<I> </I> Italic. *Kursiv stil*.
<TT> </TT> Teletype. Text med fast bredd, visas ofta med `Courier`.
<BIG> </BIG> Visar texten i stor stil.
<SMALL> </SMALL> Visar texten i liten stil.
<STRIKE> </STRIKE> och **<S> </S>** ~~Överstruken text~~.
<U> </U> <u>Understruken text</u>.

```
<B>Fet</B><BR>
<I>Kursiv</I><BR>
<TT>Teletype</TT><BR>
<BIG>Större</BIG><BR>
<SMALL>Mindre</SMALL><BR>
<STRIKE>Överstruken</STRIKE><BR>
<S>Överstruken</S><BR>
<U>Understruken</U>
```



Teckenstilelement får kombineras. För att åstadkomma text som både är fet och kursiv skriver man då <B><I>text</I></B>. Observera att koderna ska anges i ordningen ”inifrån och ut.” Att skriva <B><I>text</B></I> är inte tillåtet. Exakt hur text markerad med flera teckenstilelement samtidigt ska visas på skärmen är upp till varje webbläsare. Om det inte fungerar som du har avsett, prova med att byta ordning på koderna, d.v.s. i detta fall <I><B>text</B></I>.

Adressuppgifter

För att skriva ut adressuppgifter överst eller längst ner på en webbsida kan man använda koden **<ADDRESS> </ADDRESS>**. Text inom dessa koder visas ofta som kursiv text i ett eget stycke. Utseendet kan även påverkas med style sheets (se detta avsnitt).

Typsnittselement

Styrkoderna **<FONT>** och **<BASEFONT>** kan användas för att styra storlek, färg och typsnitt på markerad text. Det rekommenderas att man *inte* använder dessa koder, utan använder style sheets istället (se detta avsnitt).

<FONT> kan ha tre olika attribut: SIZE="n" styr textens storlek (n är antingen en siffra mellan 1 och 7 eller ett relativt värde mellan -4 och +4), COLOR="färg" styr textens färg antingen med

en färgkod i hex eller ett färgnamn (se avsnittet om färger) och FACE="typsnitt1, typsnitt2" styr typsnittet (man anger en lista på typsnitt, med det typsnitt som föredras först). Styrkoden avslutas med **</FONT>**.

<BASEFONT SIZE="n"> kan användas för att ändra normalstorleken på texten. Storleksförändringar på viss text görs då med relativa värden till <FONT SIZE> enligt ovan. <BASEFONT> anges i dokumentets <HEAD>-del.

Brytningsstreck

För att få en horisontell linje tvärs över sidan använder man styrkoden **<HR>** (horizontal ruler). Den här styrkoden klarar sig själv och ska inte ha någon slutkod.

Varianter:

<HR SIZE="n"> Anger tjockleken på linjen i antal pixlar (bildpunkter).

<HR WIDTH="n"> Anger linjens bredd i antal pixlar.

<HR WIDTH="n%"> Anger linjens bredd i procent av skärmens bredd.

<HR ALIGN="LEFT"> Anger placeringen. Även CENTER och RIGHT fungerar.

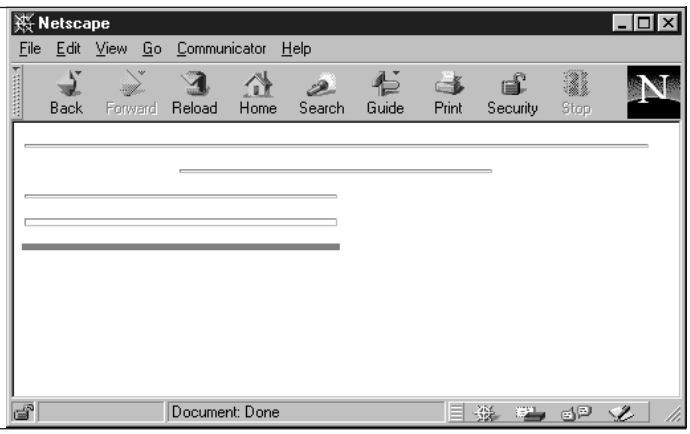
<HR NOSHADE> Strecket förlorar sin tredimensionella effekt och blir helsvart.

Du kan kombinera dem:

<HR WIDTH="30%" ALIGN="LEFT">

```
<HR>
<HR WIDTH=”50%”>
<HR WIDTH=”50%” ALIGN=”LEFT”>
<HR WIDTH=”50%” ALIGN=”LEFT”
    SIZE=”5”>
<HR WIDTH=”50%” ALIGN=”LEFT”
    SIZE=”5” NOSHADE>
```

Utseendet på olika brytningsstreck



Kommentarer

Ibland vill man infoga kommentarer i sin HTML-kod som inte ska synas på webbsidan. Om du till exempel har mycket kod så kan du få bättre överblick över den med hjälp av kommentarer. Du kan också använda kommentarer för att ge anvisningar till andra personer som ska ändra i filen senare. Kommentarer markeras med **<!--** och **-->**, och all text som läggs mellan dessa koder visas inte upp av webbläsaren.

<!-- Mina adresser -->

<!-- Lista över samtliga medlemmar -->

<!-- Fyll i månadens omsättning här -->

En annan användning för kommentarer är att snabbt ta bort stora delar av en webbsida som t.ex. är under uppbyggnad. Genom att sätta <!-- i början och --> i slutet kan man snabbt göra delar av webbsidans innehåll osynligt. Var bara uppmärksam så att den kod du på detta sätt ”kommenterar bort” inte också innehåller kommentarer. Webbläsaren kommer nämligen att avsluta bortkommenteringen vid den första --> den stöter på.

Man kan även kommentera bort enstaka styrkoder genom att sätta ett utropstecken direkt efter mindre-än-tecknet: <!KOD>. Denna kod kommer då inte att få effekt.

Observera att även om kommentarerna inte syns på själva sidan så är de inte hemliga. De syns om användaren tittar på HTML-koden via funktionen View Source eller motsvarande.

Svenska bokstäver och förbjudna tecken

Olika datortyper (PC, Macintosh, Unix etc) har olika sätt att hantera å, ä, ö och andra bokstäver med olika accenter, t.ex. é, ñ och ç. Internet och www följer en teckenstandard som kallas ISO 8859-1 (kallas även ISO Latin-1).

Alla välgjorda HTML- och WYSIWYG-editorer håller själva reda på detta och ser till att å, ä och ö blir riktiga, och om man skriver sina sidor för hand i en texteditor i Windows eller Unix blir det också rätt, eftersom Windows och Unix följer ISO 8859-1.

Om man skriver för hand i en texteditor på en dator som inte följer ISO 8859-1 (t.ex. Macintosh eller DOS) måste man själv hålla reda på att å, ä och ö blir rätt kodade. Det finns tre sätt att koda dessa ”extra” tecken. Det som oftast är enklast att skriva det tecken som är det rätta enligt ISO 8859-1. Om man gör på detta sätt på en Macintosh så kommer man alltså i sin kod se helt andra tecken istället för å, ä och ö, medan det blir rätt på själva webbsidan. Ett litet program som underlättar detta finns på http://www.algonet.se/~anders/data/mac.html

Det andra sättet, som också är vanligt, är att använda en s.k. character entity. För å, ä och ö ser det ut så här:

å **å**
Å **Å**
ä **ä**
Ä **Ä**
ö **ö**
Ö **Ö**

Observera att ”character entitys” till skillnad från HTML-styrkoder gör skillnad på stor och liten bokstav. Det går alltså inte att skriva &OUML;

Alla diakritiska tecken (tecken med accenter, prickar etc) som ingår i ISO 8859-1 har en egen character entity. En komplett lista finns på http://www.uni-passau.de/~ramsch/iso8859-1.html

Det tredje sättet är character entitys där man skriver in tecknets nummer i ISO 8859-1:s teckentabell, till exempel **A** för tecknet A.

Förbjudna tecken

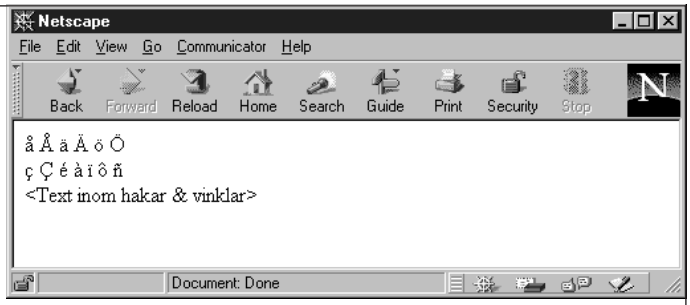
Det är inte bara å, ä, ö, é och andra diakritiska tecken som kräver speciell behandling. Till exempel så använder man ju större-än- och mindre-än-tecken för att skriva styrkoder. Och &-tecknet används för att skriva ”character entitys” (enligt ovan). Därför, om du vill ha ett större-än-, mindre-än- eller &-tecken i din text så måste du ange även detta med en ”character entity”:

<Text inom hakar **&** vinklar**>**

blir på skärmen

<Text inom hakar & vinklar>

```
&aring; &Aring; &auml; &Auml;
&ouml; &Ouml; <br> &ccedil;
&Ccedil; &eacute; &agrave;
&iuml; &ocirc; &ntilde; <br>
&lt;Text inom hakar
&amp; vinklar&gt;
```



Namngivning och webbadresser

För att webbserver och webbläsare ska veta att HTML-filen är just en HTML-fil så använder man filnamnsändelsen **.html,** d.v.s. alla HTML-filer har ett namn som slutar på **.html**, till exempel **kaka.html** eller **allan.html**. Det är viktigt att ge filerna vettiga namn, eftersom filnamnet ingår som den sista delen i sidans webbadress (se nedan). Alla användare kommer alltså att se filnamnen.

Ifall du utvecklar dina sidor på en dator med DOS eller Windows 3.x kan du inte ge dina filer ändelsen .html, så då får du istället använda ändelsen **.htm**. Det är däremot inget problem att titta på sidor med namn på .html i en webbläsare under Windows 3.x eftersom webbläsarna klarar av detta internt.

Använd aldrig å, ä eller ö i filnamnen, eftersom dessa tecken representeras olika på olika datorer. De tecken som är tillåtna är bokstäver a-z och A-Z, siffror, punkt, bindestreck och understreck.

Om webbservern går på en Unix-dator måste man dessutom skilja på stora och små bokstäver. **allan.html** och **Allan.html** är alltså två olika filer.

URL:er

Varje webbsida, bild och annan resurs tillgänglig på nätet har sin egen, unika adress, en URL (Uniform Resource Locator). Med unik menas att en adress bara kan leda till en enda sida. Samma adress kan inte leda till olika sidor. Däremot kan flera olika adresser leda till samma sida.

En URL har oftast tre delar: protokoll, datornamn och resursnamn. För webbsidor och bildfiler hämtade från en webbserver är protokollet **http** (hypertext transfer protocol). Datornamnet är namnet på den server där webbsidan ligger, till exempel **www.firma.se**. Istället för namn går det lika bra att använda serverdatorns IP-nummer, till exempel **193.44.96.160**.

Resursnamnet är namnet på den aktuella filen inklusive sökvägen till filen, till exempel **/allan/kaka.html** (d.v.s. filen **kaka.html** ligger i mappen **allan** som i sin tur ligger i ”document root”, den speciella mapp på servern där alla webbdokument ligger). I vissa fall kan filnamnet utelämnas (se avsnittet om index.html nedan). Som skiljetecken mellan mappnamn och filnamn används framåt-lutande snedstreck (/) enligt Unix-praxis.

Den fullständiga URL:en till sidan i exemplet blir **http://www.firma.se/allan/kaka.html**

URL:er skiljer i allmänhet på stora och små bokstäver. Det kan finnas URL:er, eller delar av URL:er, där stora eller små bokstäver inte spelar någon roll, men en användare har ingen möjlighet att avgöra detta, så man bör alltid utgå från att URL:er skiljer på stora och små bokstäver.

index.html

Ofta gör man ett antal webbsidor som hör ihop, och då lägger man med fördel alla HTML-filer i samma mapp. Du kan ge HTML-filerna vilka namn du vill (enligt ovan), men ”förstasidan” i mappen bör du kalla **index.html.** Det är nämligen den sidan som servern kommer att skicka om man bara anger adressen till mappen, till exempel

http://www.firma.se/

eller

http://www.firma.se/allan/

Att förstasidan ska heta just index.html är en inställning som görs i servern. Många servrar är inställda så att man även kan använda filnamn som börjar på index men slutar på andra ändelser, till exempel index.htm, index.shtml, index.cgi o.s.v. En del servrar är inställda för helt andra filnamn, t.ex. default.html, homepage.html eller welcome.html

Om man anger adressen till en mapp där det inte finns någon index-fil kommer servern, beroende på hur den är inställd, antingen att presentera en lista över alla de filer som finns i mappen (s.k. directory listing) eller ge ett felmeddelande till användaren (404 File Not Found eller 403 Forbidden).

Andra URL-delar

Förutom protokoll, datornamn och resursnamn kan URL:er ibland ha andra delar, bland annat portnummer, ankarnamn och söksträngar. Portnummer anges i de sällsynta fall då webbservern inte körs på IP-port nummer 80. Portnumret (3000 i exemplet nedan) anges skilt från datornamnet med ett kolon: **http://www.firma.se:3000/**

Ankarnamn anges efter själva URL:en med ett nummertecken (#) mellan. Se vidare avsnittet om länkar inom en webbsida. Söksträngar anges efter själva URL:en med ett frågetecken (?) emellan. Se vidare avsnitten om formulär och CGI.

Publicera dina webbsidor

Medan man utvecklar sina webbsidor arbetar man oftast med HTML-filerna på sin egen hårddisk. Man redigerar filerna med något program (enligt ovan) och tittar på hur sidan ser ut genom att öppna den i en webbläsare (till exempel Netscape) via ”Open File”, ”Open Page”, ”Open Local” eller motsvarande. Men så länge sidorna ligger på din hårddisk kan ingen annan se dem. För att andra ska kunna se dina webbsidor måste du flytta över HTML-filerna till en gemensam hårddisk som alla kan komma åt via nätet. Du måste använda en server.

En server är ett program som är igång på en speciell dator. Ofta uttrycker man sig lite slarvigt och kallar även själva datorn för server, men ska man vara noga så är en server ett program. Programmet lyssnar hela tiden efter anrop som kommer via nätet från olika klientprogram. Klientprogrammen är installerade i användarnas datorer och gör att användarna kan uttnyttja de tjänster som servern erbjuder. Netscape och de andra webbläsarna är klientprogram för www.

Server och klient använder ett protokoll, en fast uppsättning kommandon, för att kommunicera med varandra. När man publicerar webbsidor använder man en server som kan ”prata” protokollet http (hypertext transfer protocol). En sådan server kallas webbserver. Andra servrar ”pratar” andra protokoll, till exempel så finns det mailservrar och filservrar. För att komma åt dem använder man klientprogram som kan dessa protokoll, t.ex. en mailklient.

För att få sina HTML-filer från sin egen hårddisk till hårddisken på den dator som kör webbserverprogrammet loggar man in med användarnamn och lösenord och kopierar dit dem. På serverdatorns hårddisk finns en speciell mapp, ”document root”, där man lägger alla de HTML-filer och bildfiler som ska kunna nås från webben. Document root-mappen brukar heta **htdocs**, **html**, **docs**, **public_html** eller liknande.

Om vi sparar vår webbsida i en fil med namnet **kaka.html** och kopierar den till document root-mappen på www.firma.se kommer den att få webbadressen **http://www.firma.se/kaka.html**

Om vi istället skapar en mapp **allan** i document root och lägger **kaka.html** där så kommer den få webbadressen **http://www.firma.se/allan/kaka.html**

Länkar

Med ”länk” menas en referens till något annat. Man ”går” dit eller man hämtar nåt därifrån. Genom att klicka på en länk kommer användaren till en annan webbsida. En länk kan även leda till en viss plats på en sida, till en fil som ska hämtas eller till en e-postadress. Länkar på webbsidor markeras vanligen genom att texten har en annan färg (t.ex. blå) och/eller är understruken eller inramad. Även bilder (t.ex. knappar) och delar av bilder kan fungera som länkar.

Länkar görs med styrkoden **<A HREF="url">** som står för ”anchor” och ”hypertext reference”. Efter HREF skriver du adressen (URL:en) till den webbsida eller annat som länken ska leda till. Styrkoden avslutas med **</A>** och texten mellan start- och slutkod är länktexten som visas som blå och understruken i de flesta grafiska webbläsare. Icke grafiska webbläsare kan visa länkar på andra sätt.

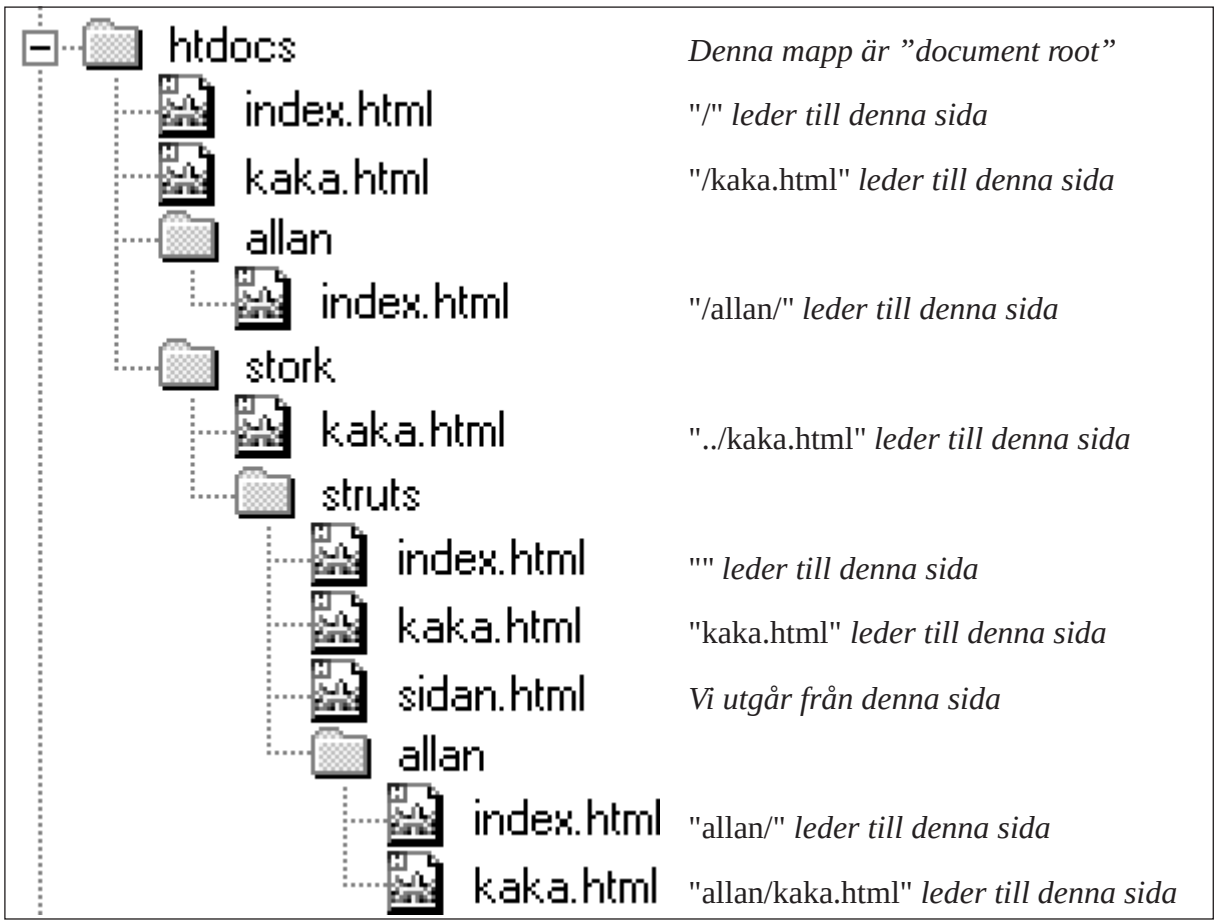
Adressen kan anges som en absolut URL (se avsnittet om webbadresser ovan). Exempel:

<A HREF="http://www.firma.se/kaka.html">Allan tar kakan</A>

Adressen kan även anges som en s.k. relativ adress. En relativ adress är ett förkortat skrivsätt som man kan använda om sidan man pekar på ligger på samma server. Man anger då bara skillnaden jämfört med adressen till den sida man länkar ifrån. Ligger sidorna i samma mapp räcker det med att ange filnamnet.

Exempel:

<A HREF="kaka.html">...</A> Filen ligger i samma mapp.
<A HREF="allan/kaka.html"> Filen ligger i en undermapp.
<A HREF="../kaka.html"> Filen ligger i en mapp uppåt i hierarkin.
<A HREF="/kaka.html"> Filen ligger på serverns översta nivå (i document root).
<A HREF="allan/"> Länken leder till indexfilen i en undermapp.
<A HREF="/allan/"> Länken leder till indexfilen i en annan mapp på översta nivån.
<A HREF="/"> Länken leder till indexfilen på översta nivån (i document root).
<A HREF=""> Länken leder till indexfilen i nuvarande mapp.



Diagrammet visar till vilka filer som en länk på sidan http://www.firma.se/stork/struts/sidan.html *kommer att leda.*

Fördelen med relativa länkar är att det är lättare att flytta innehållet i en hel mapp till en annan server (eller från din egen dator till servern). Länkarna inom mappen håller fortfarande ihop. Det tillåter också webbläsaren och servern att kommunicera mer effektivt, och inte minst, så innebär det att du inte behöver skriva lika mycket.

När webbläsaren uttyder en relativ adress utgår den ifrån den nuvarande sidans adress. Om man vill att webbläsaren ska utgå från en annan adress kan man ange styrkoden **<BASE HREF="url">** i dokumentets ”head”-del, Adressen definierad i <BASE> måste vara en absolut adress.

Länkar till mappar

När du anger en adress som leder till en mapp, tänk på att avsluta den med ett snedstreck. Skriv alltså **http://www.firma.se/** och inte bara **http://www.firma.se**

Detta gör kommunikationen mellan klient och server lite effektivare. Ifall servern får en begäran som leder till en mapp, och den inte avslutas med snedstreck, kommer servern svara klienten att sidan inte finns på angiven adress, men däremot på adressen med snedstreck på slutet. Sedan får klienten ta en ny kontakt med servern och begära om sidan.

En sak du bör vara medveten om är att länkar som leder till en mapp (d.v.s underförstått indexfilen i mappen) inte kommer att fungera medan du arbetar med HTML-filerna på din egen hårddisk. Att användaren ska få filen ”index.html” skickad till sig när hon begär ett mappnamn är en funktion i webbservern. När du tittar på sidorna lokalt kommer du istället få se en lista över innehållet i mappen (s.k. directory listing) eftersom sidan inte levereras av en webbserver.

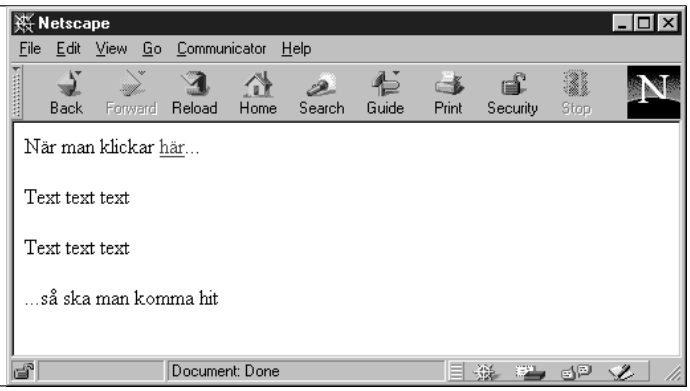
Länkar inom en webbsida

Man kan också länka till en viss plats på en sida. Detta kan vara användbart om man har en lång sida med en innehållsförteckning längst upp.

Man måste först markera texten som man ska kunna hoppa till med **<A NAME="ankare"> </A>**. Texten ”ankare” får bestå av bokstäver a-z, siffror, punkt, bindstreck och understreck. Samma ankarnamn får inte finnas på flera ställen på samma sida. Ankare kan placeras nästan var som helst i texten, men inte inne i en tabell. Glöm inte att avsluta styrkoden med </A>. En del webbläsare kan inte hantera ett ”tomt” ankare, se därför alltid till att det finns någon text mellan start- och slutkod. Den markerade texten visas inte på något avvikande sätt av webbläsaren (d.v.s. den blir inte blå som länkarna).

För att sedan göra en länk till denna punkt skriver man <A HREF="#ankare">...</A>. Om man gör en länk från en sida direkt till en viss punkt på en *annan* sida måste även sidans filnamn vara med: <A HREF="kaka.html#ankare">...</A>.

```
När man klickar
<A HREF=”#ankarpunkt”>här</A>...
<P>
Text text text
<P>
Text text text
<P>
...så ska man komma
<A NAME=”ankarpunkt”>hit</A>
```



Länkar till annat än webbsidor

Det vanligaste är att man gör länkar till andra webbsidor, men man kan även länka till stillbilder och rörliga bilder, ljud och filmer, datorprogram, Word- och Excel-dokument eller vad för slags datafil som helst.

De typer av filer som webbläsaren själv kan hantera visas direkt i webbläsarens fönster medan webbläsaren för andra typer av filer måste ta hjälp av ett annat program som kan hantera denna filtyp. Det gäller då att användaren verkligen har ett sådant program installerat.

Detta extra program kan antingen fungera som en s.k. helper application (filinnehållet visas i ett eget, separat fönster bredvid webbläsaren) eller en s.k. plug-in (filinnehållet visas i en ruta inom webbläsarens fönster).

På företagsinterna webbsidor är det vanligt med länkar som leder till MS Word-filer. Länken ser ut precis som vanligt: <A HREF="wordfil.doc">Hämta en fil!</A>.

Klickar användaren på länken så skickar servern MS Word-filen och webbläsaren anropar MS Word som får ta hand om filen. För att detta ska fungera krävs alltså att MS Word är installerat i användarens dator, och att användarens webbläsare är inställd för att anropa MS Word. Under kontrollerade former, som på en företagsintern webbsida, är detta fullt möjligt, men på en öppen sida, tillgänglig för allmänheten, kan man inte lita på att sidans besökare har sina datorer inställda så.

Mediatyper

Innan servern skickar en begärd fil så berättar den för webbläsaren vilken typ av fil den skickar, till exempel om det är en HTML-fil, en GIF-bild, en MS Word-fil etc. Filtypen anges enligt ett system som heter mediatyper (kallades tidigare MIME-typer). Mediatyper är ett sätt att innehålls-deklarera dokument som skickas över nätet.

Servern använder filnamnets sista del (.html, .gif, .doc etc) för att veta vilken mediatyp som den ska ange till webbläsaren. I servern finns en lista på vilka filnamns-ändelser som hör till vilka mediatyper. I sin webbläsare ställer man sedan in vad webbläsaren ska göra med filer av olika mediatyper, till exempel visa filen i det egna fönstret, ropa på ett externt hjälp-program eller spara på disk.

Om man lägger upp ett dokument med en ändelse som servern inte har i sin lista kommer den att skickas med fel mediatyp angiven. Det är mediatypen som styr hur webbläsaren ska tolka filen som kommer, och är den fel så kommer inte webbläsaren kunna hantera filen på rätt sätt. Detta oavsett om webbläsaren skulle ha känt igen filen med ledning av ändelsen. Därför måste man se till att servern är rätt inställd om man publicerar dokument med ovanligare filtyper.

Den enda gången som webbläsaren tar hänsyn till filnamnets ändelse är när man laddar in filer från sin egen hårddisk, annars litar den på att servern vet vad den skickar.

Länk till en e-postadress

För att göra en länk som leder till en e-postadress skriver man en <A HREF>-kod som vanligt, men adressen (URL:en) börjar med **mailto** istället för **http**.

```
<A HREF="mailto:info@firma.se">Skicka ett brev</A>
```

För att detta ska fungera krävs att användarens webbläsare även har funktioner för e-post (det har de flesta, inklusive Netscape) och att användaren fyllt i sitt eget namn och e-postadress samt namnet på sin server för utgående e-post (SMTP-server) i sin webbläsares inställningar (det kan

man dock inte vara säker på). Vill man vara säker på att ett brev verkligen går att skicka, oberoende av användarens inställningar, kan man använda en formulär-till-epost-gateway, till exempel FormMail (se http://www.worldwidemart.com/scripts/).

Länkar till ftp- och gopher-servrar, Usenet News och lokala filer

Det går också bra att göra länkar som leder till filer på andra sorters servrar, till exempel ftp- och gopher-servrar. Länkarna görs med <A HREF> som vanligt, och adressen (URL:en) börjar då med **ftp** respektive **gopher** istället för **http**.

```
<A HREF="ftp://ftp.firma.se/pub/updaters/allan-401.zip">Hämta en uppdatering</A>
<A HREF="gopher://gopher.firma.se/">Besök vår gopher-server</A>
```

För att detta ska fungera krävs att användarens webbläsare klarar av protokollen ftp och gopher (det gör de flesta, inklusive Netscape).

Länkar till diskussionsgrupper i Usenet News har adresser som börjar med **news**.

```
<A HREF="news:swnet.diverse">Följ diskussionen på nätet</A>
```

För att detta ska fungera krävs att användarens webbläsare kan hantera news (det kan de flesta) samt att användaren har tillgång till en newsserver som tar in den aktuella gruppen (det kan man dock inte vara säker på).

Länkar till filer på ditt lokala filsystem har adresser som börjar med **file**. Till ditt lokala filsystem räknas din hårddisk, eventuella disketter, löstagbara hårddiskar, cd-skivor etc som du för tillfället har monterade, samt även filservrar och nätverksdiskar du är ansluten till. Att göra sådana länkar är sällan meningsfullt, utom i vissa lägen, till exempel då alla personer i en arbetsgrupp är anslutna till en filserver eller nätverksdisk. Om alla har disken monterad som E: i Windows ser länken ut såhär:

```
<A HREF="file:///E|/sales/prognos/prog-05.xls">Försäljningsprognos för maj</A>
```

Bilder

När man ska ha bilder på en webbsida infogar man inte bilden i själva HTML-filen. Bilden lagras som en separat fil, och från HTML-filen pekar man sedan på bildfilen. Den hämtas då in av webbläsaren och placeras på sidan.

Undvik för många och för stora bilder på dina webbsidor. Det tar tid att hämta hem dem över nätet och kräver stor kapacitet hos klienten. Om du har väldigt många bilder på en sida, och webbläsaren inte har minne nog för att visa alla läggs en standard-ikon ut istället för bilden vilket kan ge ett underligt intryck av sidan.

Bildformat

Det finns en uppsjö av bildformat. De som är bäst att använda på webben är **GIF** (filerna slutar på .gif) eller **JPEG** (filerna slutar på .jpg eller .jpeg). Dessa två filformat kan visas av i stort sett alla webbläsare som kan visa bilder. Vissa webbläsare kan även visa bilder av typerna **PNG** och **XBM**.

Du kan i och för sig länka till bilder i vilket filformat som helst (se ovan), men kan webbläsaren inte hantera det kommer den inte lägga ut nåt på skärmen utan istället ropa på ett hjälpprogram.

Om användaren har ett program som kan hantera det bildformatet kommer bilden visas i det programmets fönster, och alltså inte på själva webbsidan.

För att göra egna bilder behöver du ett bildbehandlingsprogram, helst ett som kan konvertera bilder från andra filformat. Exempel på ett bra bildbehandlingsprogram är Adobe Photoshop (finns för Windows, Macintosh och en del Unix-dialekter) som är ett stort, kommersiellt program med många funktioner. Enklare ritprogram och konverteringsprogram finns att hämta från nätet som freeware och shareware. Du kan också använda bilder från clip art-samlingar. Se bara till att bilden är i GIF- eller JPEG-format.

Ett annat tips är att om du är ute och surfar och hittar en bild du vill ha, klicka med högra musknappen (Windows) eller håll musknappen nere (Macintosh) på bilden. Välj ”Save This Image as...” så får du hem bilden och kan använda den själv. Tänk dock på att bilden kan vara skyddad av upphovsrätt. Använd inte bilder som du inte har tillstånd till att använda.

Lägg in bilden på sidan

För att lägga in bilder använder man styrkoden **<IMG>**. Denna styrkod saknar slutkod, men har istället massor av attribut att välja bland. Det viktigaste attributet är **SRC** (source), som anger adressen till bildfilen. Man kan antingen ange en absolut adress (hela adressen, börjar på http://), eller också kan man ange sökväg och filnamn relativt den mapp som HTML-filen ligger i. Precis samma som för länkar alltså.

Exempel:

```
<IMG SRC="bildfil.gif">
```

Flera av attributen nedan kan kombineras, då lägger man mellanslag mellan. Exempel:

```
<IMG SRC="bildfil.gif" LOWSRC="bild.jpeg" ALT="Ett rött hjärta" BORDER="2"
WIDTH="40" HEIGHT="20" ALIGN="LEFT">
```

Alternativ text

ALT-texten är den alternativa text som skrivs ut istället för bilden när någon med en textbaserad webbläsare (t.ex. Lynx) tittar på sidan. Det är även den text som visas bredvid bildsymbolen när man laddar sidan med Netscape med bildvisning avslaget (många som sitter med långsamma modemförbindelser väljer att göra det för att spara tid). Vissa webbläsare visar också denna text under tiden som bilden laddas in eller om man för muspekaren över bilden.

Exempel:

```
<IMG SRC="bildfil.gif" ALT="Ett rött hjärta">
```

Observera att ALT-texten inte ska vara en beskrivning av bilden, utan ett *alternativ* för dem som inte ser bilden. Om du till exempel använder en liten grön cirkel som en markering, skriv då inte ALT="Liten grön cirkel", utan hellre ALT="*". Stjärnan gör samma ”jobb” i en textbaserad läsare som den gröna cirkeln gör i en grafisk. Bilder som enbart är till för dekorering kan mycket väl ha ALT="".

Som alternativ text till en firmalogotyp kan man skriva firmanamnet – inte ALT="Logotyp". Som alternativ text till bilder som illustrerar en artikel kan man skriva sånt som normalt skulle ha stått i en bildtext, ifall bilden fanns i en bok. Istället för ALT="En bild på Bohus fästning" kan man skriva ALT="Bohus fästning byggdes på 1300-talet och är idag delvis raserad".

Höjd och bredd

Med **HEIGHT="n"** och **WIDTH="n"** anger man hur stor bilden ska vara när den visas på sidan. Värdet på n räknas i pixlar (bildpunkter). Det vanligaste är att ange den storlek som bilden faktiskt har. Visserligen får bilden rätt storlek även om man inte anger något alls, men inladdningen av sidan går fortare om man anger bildens mått. När webbläsaren laddar in sidan och stöter på en <IMG>-kod kan den reservera plats på sidan och ladda in bilden senare. Om man inte har angett måtten måste inladdningen avbrytas och bilden hämtas innan inladdningen återupptas.

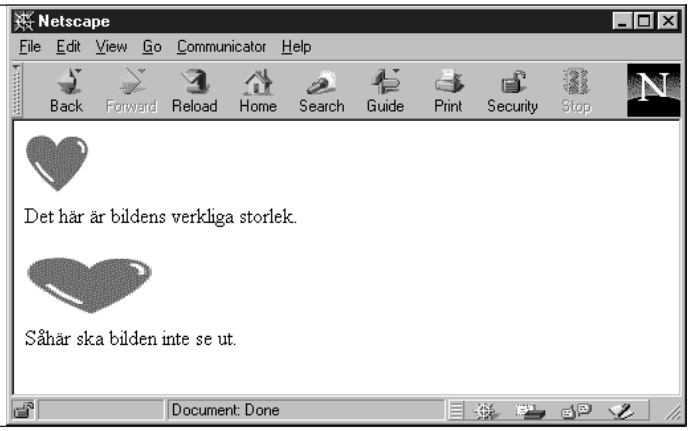
Exempel:

```
<IMG SRC="bildfil.gif" HEIGHT="45" WIDTH="90">
```

Man kan också använda WIDTH och HEIGHT för att ändra storleken på en bild genom att ange andra värden än vad bilden egentligen har. Detta rekommenderas inte, annat än för enfärgade dekorränder och liknande.

Man kan även ange storleken i procent av sidans storlek. Detta kan användas om man till exempel har en enfärgad dekorlinje som man vill ska gå över hela sidan. Med WIDTH="100%" kommer linjen alltid fylla hela bredden, oberoende av hur brett användarens fönster är.

```
<IMG SRC=”bildfil.gif”
WIDTH=”50” HEIGHT=”50”>
<BR>
Det här är bildens verkliga
storlek.
<P>
<IMG SRC=”bildfil.gif”
WIDTH=”100” HEIGHT=”50”>
<BR>
Såhär ska bilden inte se ut.
```



Alternativ bild

Med attributet **LOWSRC="URL"** kan man lägga in adressen till en bildfil som är mindre (i kilobyte räknat) än den ordinarie bildfilen, till exempel en lågupplöst, svart/vit bild, eller en hårt komprimerad JPEG-bild. Den bilden laddas in direkt när man kommer till sidan (och det kan ju gå ganska fort om bildfilen är liten).

När resten av sidan väl är laddad hämtas den riktiga versionen av bilden (som du har angivit som SRC). Besökaren får alltså en bra förhandsvisning av den riktiga bilden. Bilderna ska vara lika stora i pixlar räknat. Om de inte är det så kommer den bilden som anges som SRC skalas om till samma storlek som LOWSRC-bilden. Det går också bra att ange HEIGHT och WIDTH som då påverkar bägge bilderna.

LOWSRC accepteras bara av vissa webbläsare; i andra ger den ingen effekt.

Ram runt bilden

Styrkoden **BORDER="n"** gör att det blir en ram kring bilden med tjockleken n pixlar. Ramens färg blir densamma som omgivande text, d.v.s. svart om inget annat angetts. Anges inte BORDER blir det ingen ram.

Marginaler runt bilden

VSPACE="n" och **HSPACE="n"** används för att bestämma minimiavståndet till den omgivande texten när man figursätter text runt om en bild med ALIGN (se nedan). VSPACE reglerar det vertikala, och HSPACE det horisontella avståndet. Värdet på n räknas i pixlar.

Figursättning

Med **ALIGN="LEFT"** och **ALIGN="RIGHT"** placerar man bilden mot webbläsarfönstrets vänstra respektive högra kant. Texten figursätts runt om bilden.

ALIGN="BOTTOM" lägger bildens nederkant i nivå med textens baslinje. **ALIGN="MIDDLE"** centrerar bilden vertikalt till textens baslinje. **ALIGN="TOP"** lägger bildens överkant i nivå med toppen på den högsta tecknet på raden.

Om du har text figursatt runt om en bild, och vill börja på en ny rad, under bilden, använder du attributet **CLEAR** till koden
. Den nya raden kommer att börja precis nedanför bildens undre kant.

<BR CLEAR="RIGHT"> Flyttar efterföljande text till närmaste lediga högra marginal.

<BR CLEAR="LEFT"> Flyttar efterföljande text till närmaste lediga vänstra marginal.

<BR CLEAR="ALL"> Flyttar efterföljande text till den närmaste rad där både höger och vänster marginal är ledig. Detta kan användas när du har bilder på båda sidorna om texten och inte vet vilken av bilderna som är störst.

```
Vanligtvis hamnar en bild
<IMG SRC=”bildfil.gif”> mitt
i texten, på samma baslinje.
<P>
<IMG SRC=”bildfil.gif”
ALIGN=”LEFT”>
Om man anger ALIGN så blir
texten figursatt runt om
bilden. I detta fall ligger
bilden mot sidans vänstra kant,
och texten flödar runt om.
<BR CLEAR=”ALL”>
För att flytta ner
efterföljande text till en fri
marginal kan man använda CLEAR.
```

Klickbara bilder

Det kanske vanligaste när man gör en länk till en annan webbsida (se avsnittet om länkar) är att man markerar några ord i texten som får fungera som länk, men det går precis lika bra att låta en bild fungera som länk. Sätt bara bildreferensen mellan <A>-koderna, där texten annars skulle ha varit:

```
<A HREF="allan.html"><IMG SRC="bild.gif"></A>
```

Användaren följer länken genom att klicka någonstans på bilden. Ett vanligt användningsområde är knappar som ibland kan ge ett trevligare intryck än vanliga text som länkar.

I de flesta webbläsare kommer bilden att få en blå ram för att markera att det är en länk (ramen

blir röd eller lila när länken är följd, precis som för vanliga textlänkar).

Vill du inte ha en ram använder du BORDER="0":

```
<A HREF="allan.html"><IMG SRC="bild.gif" BORDER="0"></A>
```

Tänk bara på att utforma bilden och den omgivande texten så att man tydligt ser att bilden verkligen är en länk.

Observera att du inte bör ha något mellanrum eller radbrytning mellan koderna. Låt alltså större än- och mindre än-tecknen mötas. Gör du inte det kan du få små, blåa, oönskade streck i närheten av bilden, eftersom även mellanslagen räknas som en del av länktexten, och därför blir understrukna. Det är också extra viktigt att ange ALT-text för bilder som fungerar som länkar, annars blir sidan helt oanvändbar i textbaserade webbläsare.

Man kan också låta olika delar av bilden länka till olika sidor. Detta görs med en s.k. imagemap (se detta avsnitt).

Mer om olika bildformat

GIF står för Compuserve Graphics Interchange Format. Det är ett mycket populärt bildformat på webbsidor. Alla webbläsare som kan visa bilder kan visa GIF-bilder.

GIF-bilder lämpar sig i första hand för grafik med få färger och stora enfärgade ytor, till exempel knappar, ritningar, textplattor, dekorlinjer etc. GIF är ett s.k. dekorfärgsformat. När bilden skapas bestäms en färgpalett med alla de färger som bilden innehåller. Ju fler olika färger desto mer plats tar filen (i kilobyte räknat). Maximalt antal olika färger i en GIF-bild är 256 (8 bitar per pixel), men ofta försöker man hålla sig under 30.

En GIF-bild kan ha transparent bakgrund. Man definierar en av färgerna i färgpaletten som transparent, och då kommer webbsidans bakgrundsfärg synas igenom.

En GIF-bild kan vara ”interlaced”. Vanligtvis sparas bildinformationen linje för linje uppifrån och ner, och under tiden som bilden laddas in ritas den också upp så. Om man istället sparar bilden som ”interlaced” så sparas linjerna i bilden i en annan ordning vilket gör att bilden ser ut att smälta in (upplösningen blir gradvis bättre och bättre). Den färdiga bilden ser likadan ut på webbsidan, skillnaden är hur den ser ut medan inladdningen fortfarande pågår.

GIF-standarden medger också animering. En följd av flera gif-bilder kan slås samman till en fil och visas en efter en, ungefär som en bläddерboksfilm. Läs mer om animerade GIF-bilder på http://members.aol.com/royalef/gifanim.htm

JPEG står för Joint Photographic Expert Group. JPEG-bilder är också ett ganska vanligt bildformat på webbsidor. De flesta webbläsare som kan visa bilder kan visa JPEG-bilder.

JPEG lämpar sig i första hand för fotografier och andra bilder som innehåller många olika färger i kontinuerliga toner, till exempel hudfärg, en himmel som innehåller flera tusen olika blå nyanser etc. JPEG-formatet är ett RGB-färgsformat (d.v.s. varje färg är uppbyggd av rött, grönt och blått), och en bild kan innehålla upp till 16,7 miljoner olika färger (24 bitar per pixel).

Bilderna komprimeras med en förstörande algoritm, det innebär att den information som ögat har svårast att uppfatta slängs bort. När man sparar bilden kan man välja hur hårt denna komprimering ska ske. Vid höga komprimeringar kan klara, distinkta linjer i bilden bli suddiga. JPEG stöder inte transparenta bakgrunder, interlacing eller animering.

PNG är ett nytt format som introducerats för att vara bättre än GIF, men det är ännu inte så många webbläsare som kan visa PNG-bilder. PNG står för Portable Network Graphics. PNG kan även uttydas Png is Not Gif.

XBM står för X-Windows Bitmap, och är ett enbitars-bildformat (d.v.s. bilderna är svart/vita utan gråskalor) som används i vissa Unix-tillämpningar. XBM-bilder kan visas av Netscape, men få andra webbläsare.

Listor

Det finns tre sorters listor i HTML:

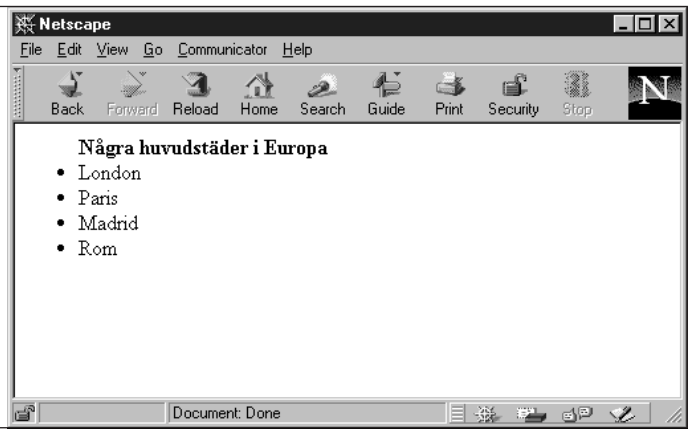
- punktlista, som denna (s.k. unordered list)
- numrerad lista (s.k. ordered list)
- definitionslista (s.k. definition list)

Listor kan staplas, d.v.s. man kan ha en lista inuti en lista. Håll bara koll på slutkoderna, annars kan det bli väldigt rörigt!

Punktlista

En punktlista görs med styrkoden **<UL>** som står för unordered list. Listan inleds med <UL> och avslutas med **</UL>**. Varje punkt på listan inleds med **<LI>** (list item). Det går att skjuta in en rubrik till listan innan första <LI>. Rubriken är inte en punkt på listan och ska alltså inte ha <LI>.

```
<UL>
<B>Några huvudstäder
i Europa</B>
<LI>London
<LI>Paris
<LI>Madrid
<LI>Rom
</UL>
```



I en grafisk webbläsare kommer listans innehåll att dras in lite åt höger. För varje <LI> kommer webbläsaren lägga ut en inledande fet punkt. Listor inuti listor får annorlunda utseende på punkterna. Den första listnivån får en fylld cirkel, nivå två får en ofylld cirkel och nivå tre får en fyrkant.

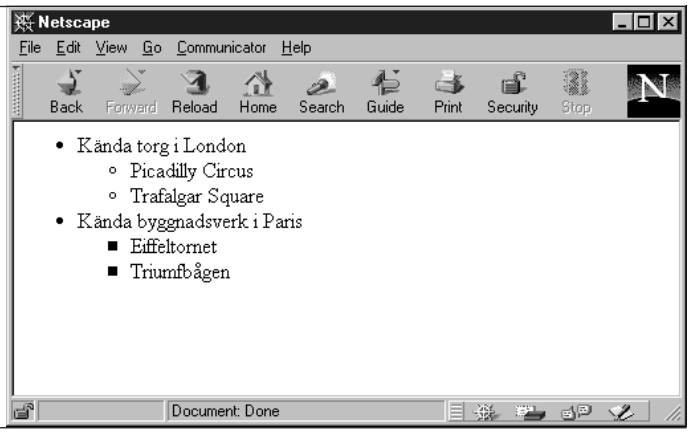
Man kan själv styra utseendet på punkterna med attributet **TYPE**. TYPE="DISC" ger en fylld cirkel, TYPE="SQUARE" ger en fyrkant och TYPE="CIRCLE" ger en ofylld cirkel. Du kan alltså få alla punkter att se likadana ut eller ändra ordningen.

Anges TYPE till <UL> gäller typen hela listan, och anges TYPE till <LI> gäller typen för en individuell listpunkt.

Andra symboler än cirklar och fyrkanter kan åstadkommas med style sheets (se detta avsnitt).

```
<UL>
<LI>Kända torg i London
<UL>
<LI>Picadilly Circus
<LI>Trafalgar Square
</UL>
<LI>Kända byggnadsverk i Paris
<UL TYPE="SQUARE">
<LI>Eiffeltornet
<LI>Triumfbågen
</UL>
</UL>
```

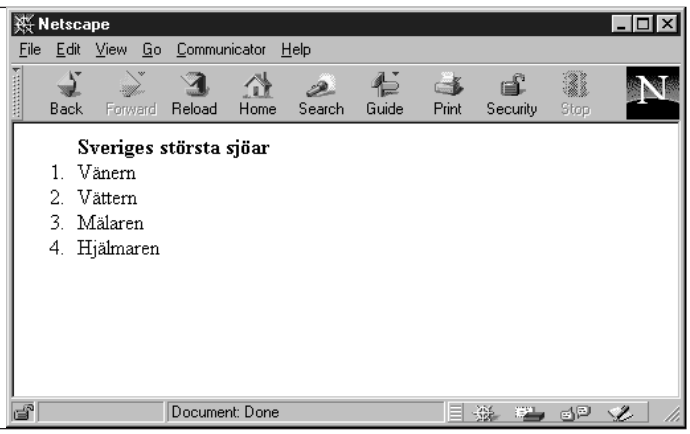
Man kan ha listor inuti en annan lista



Numrerad lista

En numrerad lista görs med styrkoden **<OL>** som står för ordered list. Samma princip som onumrerad lista: inled med <OL>, varje punkt på listan inleds med **<LI>** och hela listan avslutas med **</OL>**. Varje punkt på listan kommer automatiskt numreras i ordning.

```
<OL>
<B>Sveriges största sjöar</B>
<LI>Vänern
<LI>Vättern
<LI>Mälaren
<LI>Hjälmaren
</OL>
```



Varianter:

<OL TYPE="A"> listan numreras A, B, C, D...
<OL TYPE="a"> listan numreras a, b, c, d...
<OL TYPE="I"> listan numreras I, II, III, IV...
<OL TYPE="i"> listan numreras i, ii, iii, iv...
<OL START="4"> numreringen startar från 4. Ange alltid startvärdet som en siffra, även om du angett något annat som TYPE.
<LI VALUE="4"> denna listpunkt har värdet 4, gäller härifrån och framåt

Definitionslista

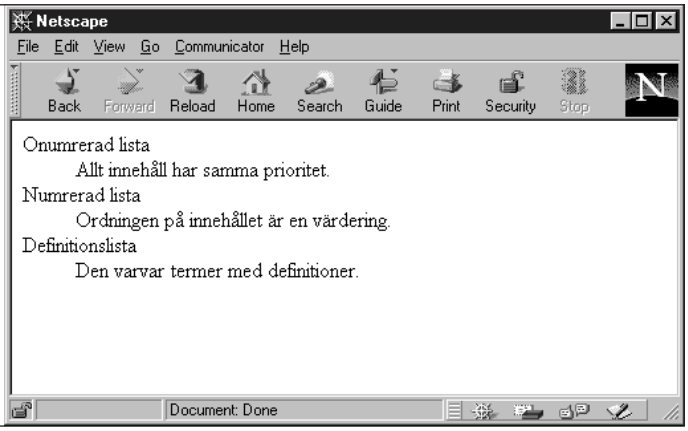
En definitionslista kan användas då man skriver ordförklaringar, uppslagssidor etc. Den här listtypen skiljer sig mot de två tidigare då du här måste varva term och definition. Börja som vanligt med en inledande styrkod **<DL>**. Sedan kommer ”termkoden” **<DT>** och första termen. Därefter ”definitionskoden” **<DD>** och definitionen. Varken <DT> eller <DD> behöver någon slutkod. Hela definitionslistan ska avslutas med **</DL>**

Den här typen av lista är speciellt användbar när man har mycket textmassa i definitionen, då blir raderna snyggt indragna. Indraget blir lika stort som för de andra listtyperna.

Termen får bara innehålla formaterad text, medan definitionsdelen kan innehålla flera stycken, separerade med t.ex. <P>.

Normalt sett hamnar term och definition på varsin rad efter varandra. Om man anger listan som **<DL COMPACT>** så sätter webbläsaren både termen och definitionen på samma rad om de får plats. Detta tillägg accepteras inte av alla webbläsare.

```
<DL>
<DT>Onumrerad lista
<DD>Allt innehåll har samma
prioritet.
<DT>Numrerad lista
<DD>Ordningen på innehållet
är en värdering.
<DT>Definitionslista
<DD>Den varvar termer med
definitioner.
</DL>
```



Äldre listtyper

Förutom <UL>, <OL> och <DL> finns det även två äldre listtyper: **<DIR>** och **<MENU>**. För båda dessa listtyper markeras listpunkterna med <LI>.

Tanken var att en <DIR>-lista skulle visas som en meny i flera spalter, och att en <MENU>-lista skulle visas som en enspaltig meny, men i praktiken visas båda dessa listor likadant som en <UL>-lista.

HTML-specifikationen rekommenderar att man *inte* använder dessa listtyper.

Tabeller

Tabeller kan användas för att presentera data i kolumner på ett snyggt sätt. Det är också vanligt att folk använder tabeller för att få större kontroll över textens placering på sidan.

En tabell inleds med styrkoden **<TABLE>** och avslutas med **</TABLE>**. Tabellens innehåll är uppdelat på en eller flera rader och varje rad är uppdelad i ett eller flera datafält. Det första man gör efter <TABLE> är att börja en ny rad med **<TR>** (table row). Sen kommer det första datafältet, som anges med **<TD>** (table data). I datafältet skriver man sedan själva texten. Man fyller sedan på med så många datafält man vill ha på den första raden. Varje datafält inleds med <TD>. Nästa rad inleds med en ny <TR> o.s.v.

Vill man kan man byta en eller flera <TD>-koder mot **<TH>** (table header). Innehållet i en sådan cell kommer att centreras och visas i fetstil, vilket kan vara lämpligt för rubriken för en kolumn, eller liknande.

Styrkoderna <TR>, <TD> och <TH> kan avslutas med </TR>, </TD> respektive </TH>, men det är inte nödvändigt. Det är underförstått att förra raden är slut när man börjar på en ny.

Ett datafält i en tabell kan innehålla i stort sett vad som helst: text, bilder, länkar, listor etc. Undvik att låta ett datafält i en tabell innehålla en ny tabell. Det är tillåtet enligt HTML-specifikationen, men det är få webbläsare som kan hantera detta.

```
<TABLE BORDER=”1”>
<CAPTION ALIGN=”TOP”>Min
 arbetsvecka</CAPTION>
<TR><TH>Måndag<TH>Tisdag
 <TH>Onsdag<TH>Torsdag
 <TH>Fredag
<TR><TD>Läser posten<TD>Kurs
 <TD>Läser posten<TD>Jobbar
 <TD>Möte
<TR><TD>Jobbar<TD>Kurs<TD>Möte
 <TD>Sen lunch<TD>Ledig
</TABLE>
```

Justera tabellens utseende

Det finns flera olika attribut till styrkoden <TABLE> som påverkar tabellens utseende (se nedan). Vill du använda flera så anger du dem efter varandra med mellanslag emellan:
<TABLE BORDER="1" WIDTH="100%" CELLPADDING="3" CELLSPACING="2">

BORDER="2" används för att ge tabellen kanter, vilket ger ett trevligt tredimensionellt utseende. Siffran anger kantens bredd och räknas i antalet pixlar (bildpunkter på skärmen). Celler som inte innehåller något (eller som bara innehåller mellanslag eller radbyten) får inga kanter. För att kringgå detta, lägg in antingen en fast radbrytning (
) eller ett fast mellanslag ( ) i cellen.

CELLSPACING="8" bestämmer avståndet mellan cellerna. Siffran räknas i antalet pixlar (bildpunkter på skärmen).

CELLPADDING="5" bestämmer avståndet från cellens innehåll (texten) till cellkanten. Siffran räknas i antalet pixlar (bildpunkter på skärmen).

WIDTH="50" och **WIDTH="50%"** används för att sätta önskad bredd på tabellen, antingen i pixlar (bildpunkter) eller i procent av sidans bredd. Tänk på att en tabell med fast bredd som ser bra ut på din skärm kan se väldigt konstig ut om användaren har valt ett annat typsnitt eller en annan storlek på texten.

ALIGN="LEFT" placerar hela tabellen i fönstrets vänstra kant, **ALIGN="CENTER"** placerar tabellen hela tabellen centrerad i fönstret, och **ALIGN="RIGHT"** placerar tabellen i fönstrets högra kant.

COLS="5" anger antalet kolumner i tabellen. Uppgiften hjälper webbläsaren att rita upp tabellen snabbare.

BGCOLOR="#00FF66" styr bakgrundsfärgen för tabellen. Se avsnittet om färger för en genomgång av färgkoderna. Det rekommenderas att man använder style sheets istället för BGCOLOR (se avsnittet om style sheets).

Överskrift/bildtext

<CAPTION> </CAPTION> används för att ge tabellen en överskrift, underskrift, rubrik eller bildtext. Den skall ligga direkt efter <TABLE> före första <TR>. Med **ALIGN="TOP"**, **ALIGN="BOTTOM"**, **ALIGN="LEFT"** eller **ALIGN="RIGHT"** bestämmer man om rubriken placeras över, under, till höger eller till vänster om tabellen.

Justera cellernas utseende

Attributen **ALIGN="LEFT"**, **ALIGN="CENTER"** och **ALIGN="RIGHT"** används för att bestämma justeringen av cellernas innehåll mot cellens vänstra kant, centrerat i cellen respektive mot cellens högra kant. Det kan göras radvis genom att attributen anges till <TR> eller för en enstaka cell genom att attributen anges till <TD> eller <TH>.

Attributen **VALIGN="TOP"**, **VALIGN="MIDDLE"** och **VALIGN="BOTTOM"** används för att bestämma den vertikala justeringen av cellernas innehåll. Det kan göras radvis genom att attributen anges till <TR> eller för en enstaka cell genom att attributen anges till <TD> eller <TH>. **<TR VALIGN="BASELINE">** anger att innehållet i alla celler på raden skall centreras vid en gemensam baslinje.

Attributet **NOWRAP** kan anges till <TD> och <TH> och gör så att cellens innehåll inte radbryts av webbläsaren för att det ska få plats i cellen. Detta kan ge väldigt breda celler. Det rekommenderas att man istället använder style sheets för att åstadkomma detta.

För att få en cell att sträcka sig över flera kolumner i tabellen anger man **COLSPAN="n"** som attribut till <TD> eller <TH>. n är en siffra som anger antalet kolumner. <TD COLSPAN="2"> gör att datafältet sträcker sig över två kolumner.

För att få en cell att sträcka sig över flera rader i tabellen anger man **ROWSPAN="n"** som attribut till <TD> eller <TH>. n är en siffra som anger antalet rader. <TD ROWSPAN="2"> gör att datafältet sträcker sig över två rader.

Bredden för en specifik cell kan anges med **<TD WIDTH="50">**. Använd detta med försiktighet, för även om det ser bra ut på din skärm, så kan texten inne i cellerna brytas väldigt fult om användaren har en annan upplösning på sin skärm. WIDTH anges i pixlar (bildpunkter) och dessa är olika stora på olika skärmar. Denna kod stöds inte av alla webbläsare.

Attributet **BGCOLOR="#00FF66"** kan anges till <TD> och <TH> och styr bakgrundsfärgen för en enstaka cell. Se avsnittet om färger för en genomgång av färgkoderna. Det rekommenderas att man använder style sheets istället för BGCOLOR (se avsnittet om style sheets).

Färger

Färger i HTML anges med namn eller med s.k. hex-koder (se nedan). De sexton vanligaste färgerna kan anges med namn och för övriga måste man ange hex-koden. Även de sexton färger som har egna namn kan anges med hex-koder. Vissa äldre webbläsare (t.ex. Netscape 2.0) kan endast tolka hex-koderna, inte färgnamnen.

De sexton färger som har egna namn är:

black	svart	"#000000"	**green**	grön	"#008000"
silver	ljusgrå	"#C0C0C0"	**lime**	ljusgrön	"#00FF00"
gray	mörkgrå	"#808080"	**olive**	mörkgul	"#808000"
white	vit	"#FFFFFF"	**yellow**	gul	"#FFFF00"
maroon	rödbrun	"#800000"	**navy**	mörkblå	"#000080"
red	röd	"#FF0000"	**blue**	blå	"#0000FF"
purple	mörklila	"#800080"	**teal**	blågrön	"#008080"
fuchsia	ljuslila	"#FF00FF"	**aqua**	ljusblå	"#00FFFF"

Hexkoder

En hex-kod är ihopsatt av ett nummertecken (#) och tre stycken tvåsiffriga hexadecimala tal i ordningen röd-grön-blå. Hexadecimal räkning är ett talsystem som går mellan 0 och 15 istället för det vanliga decimala talsystemets 0–9. Man använder de vanliga siffrorna 0–9 samt bokstäverna A–F. Varje ”siffra” i ett hexadecimalt tal kan alltså ha ett värde mellan 0 (0 i decimal räkning) och F (15 i decimal räkning). Tvåsiffriga tal kan ha ett värde mellan 00 (0 i decimal räkning) och FF (255 i decimal räkning). För varje färg kan vi alltså välja 256 olika värden (0–255).

Färgen blandas av de tre grundfärgerna rött, grönt och blått med s.k. additiv blandning, blandning av ljusstrålar. Full styrka på alla tre färgerna ger vitt, och inget av någon färg ger svart. Det är alltså motsatt blandningsprincip som i målarlådan. Se tabellen ovan för exempel.

Med 256 olika värden på tre olika färger kan man blanda fram 16,7 miljoner olika färger. Det finns speciella färgväljarprogram som översätter till hex-koder. Funktionen finns även inbyggd i en del HTML-editorer.

Säkra färger

Alla datorskärmar kan inte visa alla de 16,7 miljoner färger som kan konstrueras med hex-koder. Till exempel så kan många PC-skärmar bara visa 216 färger och många Mac-skärmar kan bara visa 256 färger. Vissa äldre färgskärmar kan visa så få som 16 olika färger.

Om man har använt en färg som skärmen inte kan visa så tar datorn antingen den som den tycker ligger närmast, eller så försöker den rastrera fram färgen (s.k. dither). Inget av dessa alternativ är speciellt aptitligt, så det kan vara bra att hålla sig till det lägsta antal färger som är vanligt, d.v.s. 216.

De färger som anges med de sexton färgnamnen ovan kan visas korrekt på de flesta skärmar som kan visa 16 färger eller fler. Det finns också listor på ”säkra” färger som visas korrekt på 216- och 256-färgersskärmar. Läs mer om vilka färger som fungerar och varför på http://www.connect.hawaii.com/hc/webmasters/Netscape.colors.html

God kontrast

En varning kan vara på sin plats här: att röd text på röd botten inte syns inses lätt, men tänk även på att andra färgkombinationer kan vara totalt oläsliga. Och även om det ser bra ut på din skärm, så kan det se hemskt ut på en skärm som inte kan visa lika många färger. Saknas den aktuella färgen i användarens dators system byter webbläsaren den mot en annan, mer eller mindre liknande färg. Tänk också på att cirka 10 procent av den manliga befolkningen är färgblind. God kontrast mellan bakgrund och text är allstå viktigt. Ett gammalt tv-tekniker-knep är att titta på sidan i svart/vitt. Är texten läsbar med skärmen i svart/vitt-läge så ser det oftast bra ut i färg också.

Färgval för hela sidan

Ett sätt att ange färginställningar som gäller hela sidan är att ange ett antal attribut till styrkoden <BODY>. Man kan ange enfärgad eller mönstrad bakgrund, och färg för text och länkar. Framöver, när fler webbläsare stöder det, bör man *inte* använda detta sätt att ange färginställningar, utan istället använda style sheets (se detta avsnitt).

Notera att användaren kan ange andra färger i sin webbläsares inställningar, som då får företräde framför de färger du har valt.

BGCOLOR="#00FF66" Anger sidans bakgrundsfärg.
TEXT="#00FF66" Anger färgen på all vanlig text på sidan.
LINK="#00FF66" Anger färgen på alla ej besökta länkar på sidan.
VLINK="#00FF66" Anger färgen på alla besökta länkar på sidan (visited link).
ALINK="#00FF66" Anger färgen i det ögonblick man klickar på länken (activated link).
BACKGROUND="bild.gif" Anger en bakgrundsbild som repeteras över sidan. Se till att bilden är lämpad för detta, t.ex. att mönstret går skarv i skarv. Vanliga bakgrundsbilder är ljusa mönster som till exempel kan ge intrycket av att sidan är skriven på gammalt linnepapper eller en träskiva. Använd inte den här funktionen om texten blir svårläst. Det krävs en mycket diffus och svag bild för att den ska fungera som bakgrund. Är du tveksam, använd inte bilden. Ju större bild desto längre tid tar det att hämta bilden över nätet, och ju mindre bild desto längre tid tar det för användarens dator att rita upp sidan.

Vill man ange flera av dessa tillval så anger man dem efter varandra med mellanslag emellan:
<BODY BGCOLOR="#00FF66" TEXT="#000000" LINK="#0000FF">

Om man anger TEXT och LINK bör man även ange BGCOLOR, ALINK och VLINK. Om man inte gör det kan texten bli svår eller omöjlig att läsa hos de användare som har ställt in egna bakgrunds- och länkfärger i sina webbläsare, eftersom man riskerar att få t.ex. röd text på röd bakgrund.

Färgval för delar av sidan

Ett sätt att ange färginställningar för vissa bitar av texten är att ange COLOR som attribut till <FONT> (se avsnittet om strukturerad text). Det gamla sättet för att bestämma en viss bakgrundsfärg för en tabell eller en cell i en tabell är att ange BGCOLOR som attribut till styrkoderna <TABLE> respektive <TD>.

Framöver, när fler webbläsare stöder det, bör man *inte* använda detta sätt att ange färginställningar, utan istället använda style sheets (se detta avsnitt).

God smak på webben

HTML är ett kodspråk som möjliggör att webben är plattformsoberoende. Det betyder att alla oberoende av program- och hårdvara har tillgång till samma information. Med HTML strukturerar (markerar) man dokumentets innehåll i rubriker, stycke, listor eller tabeller.

Dessa markeringar (styrkoder) tolkas av en webbläsare och dokumentet presenteras på det sätt som det strukturerades. Hur sedan dokumentet *ser ut* beror helt och hållet på de inställningar (typsnitt, färger, skärmbredd, skärmupplösning o.s.v.) som läsaren har. Har man t.ex. en stor skärm breder innehållet ut sig, har man en liten skärm blir det smalare. Dokumentets layout anpassar sig till rådande förhållanden.

Att försöka manipulera layouten och tvinga besökaren att titta på dokumentet på ett visst sätt kommer oftast att leda till problem. HTML är *inte* ett sidbeskrivningsspråk, d.v.s. språket beskriver inte hur sidan ska se ut. Språket beskriver sidans struktur. Man kan sedan rekommendera olika utseende för de olika strukturerna med hjälp av style sheets. Trots det kan webbsidan se helt olika ut beroende på vilken dator och med vilket program den visas.

När man designar webbsidor måste man därför alltid tänka på att sidan ska se bra ut på så många olika system som möjligt. Om man vill att alla ska kunna se ens sidor så får man undvika koder och konstruktioner som t.ex. Netscape är ensamma om. Ju färre antaganden man gör om läsarnas programvara, skärmstorlek och datortyp man gör, desto bättre.

Stora bilder

Man ska också undvika alltför stora bilder (i kilobyte räknat), speciellt på titelsidorna. Bilder tar lång tid att föra över på långsamma modemförbindelser och lång tid att rita upp på skärmen för äldre datorer.

Tänk också på att bara cirka 10 procent av användarna rullar neråt på sidan, ifall sidan inte får plats i webbläsarens fönster. Lägg alltså det viktigaste överst, och inte en stor logotyp som täcker större delen av skärmen.

Måste du ha en stor bild, lägg den gärna på en separat sida, och ange storleken i kilobyte vid sidan av länktexten, så får besökaren avgöra om hon vill hämta hem bilden eller inte. Ange alltid WIDTH, HEIGHT och även ALT till alla bilder. Det gör att sidan laddas in fortare. Tänk också på att bilder med många färger i kan se fruktansvärda ut på äldre skärmar som inte kan visa så många färger.

Intranät

När man gör webbsidor för internt bruk inom ett företag kan man vara lite mer vårdslös eftersom man vet vilket webbläsarprogram som används, vilken typ av datorer personalen har och hur de är inställda, men fortfarande vet man inte vilken storlek på texten användaren har ställt in, vilken fönsterstorlek hon har för tillfället eller om hon valt att slå av bildvisning.

Dessutom kan det hända att vissa eller alla av någon anledning byter webbläsarprogram i framtiden, och då måste alla sidor som utnyttjar t.ex. Netscape-specifika koder justeras. Skriver man rätt från början så slipper man det extrajobbet.

God design

God designs uppgift är att göra texten lätt att läsa. God design märks inte, eftersom idealet är att läsprocessen ska gå automatiskt, utan att man tänker på det. Man ska aldrig behöva anstränga sig för att läsa texten.

Allt i designen som stör läsandet ska därför undvikas. Att välja andra bakgrundsfärger än vitt eller någon ljus färgton är dumt eftersom texten då blir svår att läsa. Mönstrade bakgrundsbilder är ännu värre, och väljer man att använda en sådan så måste man vara väldigt noggrann med att testa sidan på olika datorer och med olika skärminställningar för att vara säker på att den inte blir svårläst. Kom ihåg att du inte kan utgå ifrån att alla ser din sida som den ser ut på din skärm.

På webben ska designen också hjälpa besökarna att navigera bland sidorna. Tydliga bilder och förklarande länktexter är ett krav. Det ska klart framgå vad som händer och vart man kommer när man klickar på en länk. Det är därför också viktigt att namnge sina mappar och HTML-filer så att det av URL:erna framgår vad de handlar om.

God design har mycket med vana att göra; det är lättast att läsa text och hitta bland sidor som är uppställda på det sätt man är van vid. Därför ska man till exempel inte ändra färgen på länkarna i onödan. Länkar till sidor som användaren inte har sett är blå, och länkar till besökta sidor är lila eller röda. Ändrar man på detta så förvirrar man sina besökare.

Frames

En annan sak som stör det normala sättet att navigera bland webbsidor är att dela upp sidorna i Frames (mer om detta i ett kommande avsnitt). Användaren kan inte förutse i vilken ruta som länkade sidor kommer att hamna, URL:er pekar fel, och utskrifter blir svårare att göra. Frames-indelade sidor går inte att sätta bokmärken på, eftersom man då kommer tillbaka till en annan version av Frameset:et.

Back-knappen fungerar också olika i olika versioner av de populära webbläsarna. I en del följer man det spår av sidor i de olika rutorna som man faktiskt har sett, i andra hoppar man ur hela Frameset:et. Dessutom tar de olika rutorna upp värdefull plats på skärmen. Använd därför Frames endast då det verkligen tillför något till din sida. Och glöm inte att erbjuda ett <NOFRAMES>-alternativ för dem som inte kan eller vill se Frames.

Rörliga element

Det mänskliga seendet är konstruerat så att om någonting i synfältet rör sig så får det all uppmärksamhet. Eventuell text och andra saker på sidan kommer knappt att märkas alls. Därför ska man inte ha saker på sin sida som rör sig, om inte avsikten är att dra all uppmärksamhet dit, till exempel vid omedelbar livsfara. Rörliga element inkluderar animerade GIF-bilder, rullande texter och blinkande text.

Formuleringar att undvika

Vissa webbdesigners som har gjort sidor som bara ser vettiga ut i en viss webbläsarversion eller med en viss skärmstorlek försöker ibland skyla över detta genom att uppmana sina besökare att installera samma webbläsare eller ställa in sina skärmar likadant. Gör inte så. Dels är det oartigt att tala om för andra vilken webbläsare de ska använda, dels kanske besökaren inte har möjlighet (eller motivation nog) att byta.

Andra formuleringar som ska undvikas är ”klicka här” (det är uppenbart att man ska klicka på en länk, om man vill) och ”under konstruktion”. Alla sidor är alltid under konstant omarbetning, det behövs ingen speciell annonsering. Och är sidan inte tillräckligt klar för att visas upp alls, så ska den heller inte visas upp alls.

Var noggrann

Se alltid till att dina sidor är uppdaterade och korrekta, det gäller både textinnehåll och länkar. Gör gärna upp en plan för uppdateringen redan när du gör sidan första gången. Vissa typer av sidor (referensinformation t.ex.) uppdateras aldrig eller mycket sällan, andra ska uppdateras ofta. Och kontrollera då och då att dina länkar fortfarande leder någon vart.

Var noga med upphovsrätten. I den digitala världen är det väldigt lätt att kopiera andras texter och bilder, men lättheten gör det inte lagligt. Att kopiera andras dokument kallas för stöld, även om det sker med ett par musklick. Den som gjorde jobbet ska ha uppskattning och ingen annan.

Slutligen så ska man naturligtvis vara noga med att HTML-koden är korrekt. Glöm inte att stänga alla citat, skriva ut alla nödvändiga slutkoder och att göra svenska bokstäver och specialtecken riktigt. En del HTML-editorer har inbyggda funktioner för syntaxkontroll, det finns också speciella webbsidor på nätet som kontrollerar koden på de sidor man begär. Ett enkelt test är också att titta på koden med Netscapes ”View page source” – har du glömt att stänga citat eller gjort liknande syntaxfel kommer den sektionen av din kod att blinka (gäller Netscape 2.0 och senare). Sådana fel klarar man sig i regel ifrån ifall man använder en HTML-editor, men dessa låter dig istället göra andra typer av fel, till exempel att länkar och bildreferenser leder till lokala filer på din hårddisk (URL:en börjar på file://) vilket gör att länkarna fungerar för dig, men inte för någon annan.

Testa gärna sidorna på flera olika datorer, uppkopplade till Internet på olika sätt (modem, ISDN, fast förbindelse) och med flera olika programvaruversioner.

Style Sheets

Med style sheets kan man i detalj kontrollera sina webbsidors utseende. Man kan styra textstorlek, radavstånd, marginaler, indrag, typsnitt, text- och bakgrundsfärger, ramar runt texten etc. Stilanvisningarna kan placeras i en separat fil som då gäller för alla eller vissa av ens webbsidor, eller inkluderas i själva HTML-filen så att de gäller enbart för den sidan – hela sidan eller en viss del av sidan.

Observera att det fortfarande är så att man inte kan veta exakt hur det ser ut i varje webbläsare, eftersom alla datorsystem ser olika ut, med olika skärmstorlekar, färgmöjligheter, installerade typsnitt etc. Style sheets är en rekommendation som användaren, och användarens datorsystem, kan välja att följa helt, delvis eller inte alls.

Med style sheets kan man ange olika stilanvisningar för olika situationer. Till exempel kan en sida gjord med ett typsnitt och ett färgval som passar för visning på skärmen ha ett alternativt style sheet som gäller vid utskrift. Detta style sheet kan innehålla helt andra stilanvisningar, speciellt anpassade för att utskriften ska bli så bra som möjligt.

På samma sätt kan man ange olika style sheets för om webbsidan ska visas av en grafisk webbläsare på en stor skärm, av en textbaserad webbläsare med en liten, handhållen skärm, av en WebTV eller om den ska läsas upp av en talsyntesbaserad webbläsare.

Varje enskild användare kan också specifiera ett eller flera egna style sheets som passar just hennes datorsystem, synskärpa och smak. Stilanvisningarna i detta kan antingen alltid ges företräde framför dokumentets stilanvisningar, eller användas när en webbsida saknar egna stilanvisningar

Cascading style sheets

CSS eller CSS1 (Cascading Style Sheets, level 1) är ett enkelt men kraftfullt ”språk” för att ange stilanvisningar för webbsidor. Det finns flera olika andra style sheet-språk som har delvis olika användningsområden, och HTML-specifikationen medger att man använder vilket som helst, men alla webbläsare stöder inte alla språk.

CSS passar för de flesta allmänna och många specialiserade tillämpningar, och det stöds av flera webbläsare. Netscape Communicator stöder CSS från och med version 4 (inte fullt ut dock) och Microsoft Internet Explorer stöder CSS i begränsad omfattning i version 3 och nästan fullt ut från och med version 4. Andra webbläsare som stöder CSS är Arena, Emacs och Tamaya.

Det heter *cascading* style sheets eftersom flera style sheets kan påverka sidans slutgiltiga utseende. Man kan t.ex. ha ett länkat style sheet som gäller för hela sin webbplats och ett style sheet som gäller just den aktuella sidan. Sidan ”ärver” då anvisningarna från det länkade style sheetet, som sedan kan modifieras eller specifieras ytterligare för den aktuella sidan.

Stilanvisningar för en viss webbsida

För att ange ett style sheet för ett visst dokument använder man styrkoden **<STYLE>** som sätts i HTML-filens ”head”-del. Här kan man ange stilanvisningar för *alla* förekomster av en speciell HTML-kod, till exempel alla <H1>-koder, alla stycken definierade med <P> eller alla <LI>-listor.

Detta exempel gör att alla <H1>-rubriker på sidan får centrerad, röd text:

```
<HEAD>
<STYLE TYPE="text/css">
H1 { color: red; text-align: center; }
</STYLE>
</HEAD>
```

Notera attributet **TYPE="text/css"** som anger att det är just CSS som används. Mellan <STYLE> och slutkoden **</STYLE>** listar man stilanvisningar för alla HTML-koder som man vill påverka. Stilanvisningarna sätts inom klamrar (”krullparenteser”) och har formen av **namn: värde**. Flera stilanvisningar i samma klammer separeras med semikolon.

Vissa äldre webbläsare som inte stöder style sheets kommer att hoppa över <STYLE>-koden och därmed visa stilanvisningarna på själva webbsidan. För att undvika detta kan man dölja stilanvisningrna med kommentarskoder:

```
<STYLE TYPE="text/css"> <!--
H1 { color: red; text-align: center }
--> </STYLE>
```

Behovet av att gömma stilanvisningarna på detta sätt är dock inte så stort eftersom även många webbläsare som inte stöder style sheets (t.ex. Netscape 2.0 och 3.0) själva klarar att dölja koderna – trots att de i övrigt inte stöder style sheets.

Om man vill ha kommentarer i ett style sheet så avgränsas de med /* och */:

```
<STYLE TYPE="text/css"> <!--
H1 { color: red; text-align: center } /* här fixar jag utseendet på rubriken */
--> </STYLE>
```

Stilanvisningar för grupper

Man kan också ange stilanvisningar för *vissa* förekomster av en HTML-kod, grupperade med attributet **CLASS.**

Detta exempel gör att de <H1>-rubriker som hör till klassen ”allan” får centrerad, röd text:

```
<HEAD>
<STYLE TYPE="text/css">
H1.allan { color: red; text-align: center; }
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="allan">Denna rubrik påverkas av stilanvisningen
<H1>Denna rubrik påverkas inte av stilanvisningen
</BODY>
```

Kodens namn och klassens namn separeras med punkt: **H1.allan** i exemplet ovan. Om man låter flera olika typer av HTML-koder ingå i samma klass, så kan man adressera dem med bara klassnamnet, med en punkt framför:

```
<HEAD>
<STYLE TYPE="text/css">
.allan { color: red; }
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="allan">Denna rubrik är röd
<B CLASS="allan">Dessa tre ord</B> är röda och feta.
</BODY>
```

Gruppering och ärftlighet

Om man skriver flera HTML-koder med komma emellan så gäller stilanvisningarna för alla dessa element:

```
H1, H2, B, UL.kaka { color: red; }
```

All text som *antingen* är markerad med <H1>, <H2>, <B> eller <UL CLASS="kaka"> kommer att bli röd.

Om man inte skriver komma emellan så påverkas bara den text som är markerad med *alla* de angivna koderna *i den ordningen*.

```
H1, B { color: blue; }
H1 B { color: red; }
```

Alla <H1>-rubriker och all <B>-markerad text blir blå, utom då en del av en <H1>-rubrik dessutom är markerad med <B>. Då blir den röd:

```
<H1>Blå blå blå</H1>
Text text <B>blå blå</B> text text
<H1>Blå blå <B>röd</B> blå blå</H1>
<B><H1>Blå blå blå</H1></B>
```

Notera att text som markerats i omvänd ordning (först <B> och sedan <H1>) inte blir röd.

Text markerad med flera olika koder ärver de egenskaper som inte motsäger varandra.

```
B { text-size: 120%; }
SPAN.viktig { text-size: 120%; }
```

Text som är markerad med <B> blir 20 procent större, och om texten dessutom markeras med <SPAN CLASS="viktig"> blir den ytterligare 20 procent större.

Stilanvisningar för enstaka element

Vill man ange stilanvisningar för ett enstaka HTML-element kan man identifiera det med attributet **ID**.

Detta exempel gör att den <H1>-rubrik som har ID="kaka" får centrerad, röd text:

```
<HEAD>
<STYLE TYPE="text/css">
H1#kaka { color: red; text-align: center; }
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="allan">Denna rubrik påverkas inte av stilanvisningen
<H1 ID="kaka">Denna rubrik påverkas av stilanvisningen
<H1>Denna rubrik påverkas inte av stilanvisningen
</BODY>
```

Kodens namn och id-namnet separeras med nummertecken: **H1#kaka** i exemplet ovan. Bara ett enda element på varje sida får ha samma ID. Samma minnesutrymme används för ID och NAME, vilket gör att det på en sida inte får finnas ett ID och ett NAME med samma namn tilldelat.

Det går att ange stilanvisningar för ett enstaka element direkt i elementets styrkod. För att ange att en viss rubrik ska få röd, centrerad text kan man skriva **<H1 TYPE="text/css" STYLE="color: red; text-align: center;">**. Här anges alltså STYLE som ett attribut med ett värde av samma typ som sätts inom klamrar ovan. Att göra så kan vara motiverat i vissa fall, men observera att man då förlorar en del av styrkan och poängen med style sheets.

Stilanvisningar för pseudoklasser

Objekt som inte existerar som egna HTML-element, till exempel alla icke följda länkar på sidan eller första raden i varje stycke, kallas för pseudoklasser, och man kan även ange stilanvisningar för dessa. Kodens namn och namnet på pseudoklassen separeras med kolon:

```
A:link { color: blue; }
A:visited { color: red; }
A:active { color: green; }
```

I detta exempel blir alla oföljda länkar blå, följda länkar blir röda och länkarna blir gröna i det ögonblick man klickar på dem.

Andra pseudoklasser, som bara stöds av vissa webbläsare, är first-line och first-letter:

```
P:first-letter { font-size. 200%; }
```

Första tecknet i alla stycken blir dubbelt så stort.

```
P.allan:first-letter { color: purple; }
```

Första tecknet i stycken som hör till klassen ”allan” blir lila.

```
P:first-line { text-transform: upper-case; }
```

Första raden i varje stycke blir satt med stora bokstäver.

Utnyttja <DIV> och <SPAN>

Visserligen kan i stort sett varje styrkod i HTML ingå i en klass eller ges ett id-namn, men det finns två HTML-koder som är speciellt användbara eftersom de i sig själva inte anger något speciellt utseende, nämligen block-grupperingskoderna <DIV> och <SPAN>.

Kombinerar man dem med style sheets kan man åstadkomma vilka effekter som helst. Till exempel kan man med <SPAN> få de första orden i varje stycke med blå kapitäler:

```
<HEAD>
<STYLE TYPE="text/css">
SPAN.kap { color: blue; font-variant: small-caps; }
</STYLE>
</HEAD>
<BODY>
<P><SPAN CLASS="kap">De första</SPAN> orden i stycket är i blå kapitäler.
</BODY>
```

Stilanvisningar för olika medier

Om man vill att olika stilanvisningar ska gälla för olika medier, till exempel för visning på skärm och för utskrift, anger man attributet **MEDIA** i <STYLE>-koden.

I detta exempel blir alla <H1>-rubriker centrerade både på skärmen och vid utskrift. På skärmen blir de dessutom blå. Om sidan läses upp av en talsyntesbaserad webbläsare är stilen helt annorlunda.

```
<HEAD>
<STYLE TYPE="text/css" MEDIA="screen, print">
H1 { text-align: center; }
</STYLE>
<STYLE TYPE="text/css" MEDIA="screen">
H1 { color: blue; }
</STYLE>
<STYLE TYPE="text/acss" MEDIA="speech">
H1 { cue-before: url(bell.aiff); cue-after: url(dong.wav); }
</STYLE>
</HEAD>
```

Möjliga värden för MEDIA-attributet är **screen** (texten visas på en icke sid-indelad skärm), **print** (texten skrivs ut på icke genomskinligt material), **projection** (texten visas med en projektor), **braille** (texten visas på en punktskriftsvisare), **speech** (texten läses upp med talsyntes) och **all**

(gäller alla medier). Flera värden kan anges med komma emellan. Anges attributet inte så antas screen.

Det är också möjligt att tillverkare av webbläsare för speciella förutsättningar, till exempel för små handhållna datorer, definierar egna värden för MEDIA-attributet.

Stilanvisningar för en grupp webbsidor

Stilanvisningarna kan sparas i en separat fil som anropas från den aktuella webbsidan. På detta sätt kan man använda samma stilanvisningar för en stor eller liten grupp webbsidor, till exempel alla sidor på ett företags hela webb.

Vill man i ett senare skede förändra stilen på sidorna så gör man det snabbt och enkelt genom att man bara ändrar i den separata filen där stilanvisningarna finns. Alla sidor som länkar till stilanvisningsfilen kommer då att ändras på en gång, utan att man behöver redigera dem alls.

För att länka till ett externt style sheet använder man styrkoden **<LINK>** som sätts i HTML-filens ”head”-del. Exempel:

<LINK HREF="min-stil.css" TYPE="text/css" REL="stylesheet">

HREF pekar på en style sheet-fil som enbart innehåller stilanvisningar (som i exemplen ovan, men utan <STYLE>-koder runt).

Man kan ange en eller flera style sheet-filer med <LINK> samtidigt som man anger stilanvisningar med <STYLE>. Sidan kommer att följa stilanvisningarna både i de angivna yttre filerna och i HTML-filen. Detta är speciellt användbart om man har en style sheet-fil med företagets övergripande stilregler, och sedan vill komplettera dem med några specialregler för en eller flera webbsidor.

Om man anger olika anvisningar för samma objekt kommer den som anges eller anropas sist i HTML-filen att gälla. Exempel:

```
<LINK HREF="firmaprofil.css" TYPE="text/css" REL="stylesheet">
<LINK HREF="min-egen-stil.css" TYPE="text/css" REL="stylesheet">
<STYLE TYPE="text/css" MEDIA="screen, print">
H1 { color: black; }
</STYLE>
```

Om H1-rubriker är röda och vänsterställda i ”firmaprofil.css” och blå och centrerade i ”min-egen-stil.css” kommer de bli svarta och centrerade på just denna sida, eftersom svart färg angavs i det sista av de tre style sheets som påverkar sidan, och centrering angavs efter vänsterställning.

```
<HTML>
<HEAD>
<TITLE>Min fina webbsida</TITLE>
<STYLE type=”text/css”>
BODY { background: white; color: black; }
A:link { color: blue; }
A:visited { color: red; }
A:active { color: green; }
</STYLE>
</HEAD>
<BODY>...</BODY>
</HTML>
```

Detta stylesheet ger webbsidan vit bakgrundsfärg och gör brödtexten svart. Länkarna är blå, besökta länkar är röda och länkarna blir gröna i det ögonblick man klickar på dem.

Stilanvisningar i CSS1

Dessa stilangivelser ingår i Cascading Style Sheets, level 1. Notera att alla stilangivelser inte får effekt i alla webbläsare. En lista över vilka stilangivelser som stöds av olika versioner av Netscapes och Microsofts webbläsare finns på http://www.webreview.com/guides/style/

Se avsnittet ovan för information om hur stilangivelserna anropas från HTML-koden.

Textstil

font-family anger typsnittet. Flera namn (i prioriteringsordning) separeras med komma, och namn som innehåller mellanrum sätts inom citattecken. Förutom typsnittsnamn kan man ange fem generiska namn: **serif** (typsnitt med ”klackar”), **sans-serif** (typsnitt utan ”klackar”), **cursive** (skrivstilstypsnitt), **fantasy** (fantasistilar) och **monospace** (typsnitt med fast breddsteg). Man bör alltid ange ett av dessa, ifall användaren inte har det önskade typsnittet installerat.

```
BODY { font-family: palatino, "new century schoolbook", serif }
```

font-style kan vara **normal**, **italic** eller **oblique**. Italic ger kursiv stil, medan oblique ger en lutande stil. Detta ger ibland samma resultat.

```
CITE { font-style: italic }
```

font-variant kan vara **normal** eller **small-caps**. Small-caps ger kapitäler (ser ut som stora bokstäver, men är i de små bokstävernas höjd). System som inte kan visa kapitäler kan välja att visa versaler istället.

```
P:first-line { font-variant: small-caps }
```

font-weight styr textens vikt och kan anges absolut med **normal** eller **bold**, relativt tidigare stil med **bolder** eller **lighter**, eller med siffervärdena **100**, **200**, **300**, **400**, **500**, **600**, **700**, **800** och **900**. Normal motsvarar 400 och bold motsvarar 700. Finns typsnittet i flera olika vikter mappas de till lämpliga värden.

font-size styr textens grad (storlek) och kan anges som en absolut storlek (**xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**), som en relativ storlek (**larger**, **smaller**), i procent eller med måttenheter (t.ex. **pt**, se avsnittet om måttenheter nedan). Man bör helst inte använda måttenheter, eftersom olika människor föredrar olika textstorlek (beroende på syn, skärmstorlek och skärmupplösning). Ange hellre i procent.

```
H1 { font-size: 150% }
```

font kan styra alla textstilsattribut ovan. 12pt/14pt i exemplet nedan betyder att graden (storleken) är 12 punkter och kägeln (radavståndet) är 14 punkter.

```
P { font: 12pt/14pt bold Helvetica, Gill, sans-serif }
```

Färg och bakgrund

color anger textens eller objektets färg. Den kan anges med färgnamn (enligt avsnittet om färger ovan) eller med RGB-värden med tresiffriga hexkoder, sexsiffriga hexkoder, decimaltal eller procentvärden. Alla dessa exempel ger röd färg:

```
EM { color: red }
EM { color: #f00 }
EM { color: #ff0000 }
EM { color: rgb(255,0,0) }
EM { color: rgb(100%,0%,0%) }
```

background-color anger bakgrundsfärgen för raden, sidan eller objektet. Färgen anges som för color ovan eller som **transparent**.

```
BODY { background-color: red }
H1 { background-color: yellow }
```

background-image kan anges med url:en till en bildfil eller som **none**.

```
BODY { background-image: url(marmor.gif) }
P { background-image: none }
```

background-repeat kan vara **repeat**, **repeat-x**, **repeat-y** eller **no-repeat** och styr hur bakgrundsbilden repeteras över sidan.

background-attachment kan vara **scroll** eller **fixed** och styr ifall bakgrundsbilden följer med texten när man rullar sidan eller om bilden ligger still.

background-position styr hur bakgrundsbilden placeras och kan anges som två procentvärden eller två längdmått (för x-axel och y-axel). Man kan även ange placeringen med **top**, **center**, **bottom** respektive **left**, **center**, **right**.

```
BODY { background-position: right top }
BODY { background-position: top center }
BODY { background-position: 40% 70% }
BODY { background-position: 14cm 0cm }
```

background kan styra alla bakgrundsattribut ovan.

```
BODY { background: red url(marmor.gif) repeat scroll }
```

Textegenskaper

word-spacing styr avståndet mellan orden i texten och kan vara **normal** eller anges med måttenheter. Vid utsluten text (rak vänster- och högermarginal) kan avståndet variera om word-spacing är satt till normal. Webbläsare kan välja att ignorera denna kod.

```
H1 { word-spacing: 0.4em }
```

letter-spacing styr avståndet mellan bokstäverna i texten och kan vara **normal** eller anges med måttenheter. Vid utsluten text (rak vänster- och högermarginal) kan avståndet variera om letter-spacing är satt till normal. Webbläsare kan välja att ignorera denna kod.

```
H1 { letter-spacing: 0.1em }
```

text-decoration kan vara **none**, **underline**, **overline**, **line-through** eller **blink**.

vertical-align styr textens eller objektets placering i höjdled. Möjliga värden är **baseline**, **sub**, **super**, **top**, **text-top**, **middle**, **bottom** och **text-bottom**. Placeringen kan även anges i procent.

text-transform gör att texten växlar mellan versaler (stora bokstäver) och gemena (små bokstäver). Möjliga värden är **capitalize** (första bokstaven i varje ord blir versal), **uppercase** (all text blir med versaler), **lowercase** (all text blir med gemena) eller **none** (texten förändras inte).

text-align styr textens justering i sidled. **left** vänsterställer texten, **right** högerställer texten, **center** centrerar texten och **justify** åstadkommer utsluten text (rak vänster- och högermarginal). Alla webbläsare kan inte visa utsluten text.

```
P.utsluten { text-align: justify }
```

text-indent åstadkommer ett indrag på första raden i ett stycke och kan anges med måttenheter eller i procent.

```
P { text-indent: 1em }
P { text-indent: 10% }
```

line-height styr kägeln (radavståndet) och anges med en siffra som multipliceras med teckengraden, i procent av teckengraden, med måttenheter eller som **normal**.

```
DIV { font-size: 10pt; line-height 1.2 }
DIV { font-size: 10pt; line-height 120% }
DIV { font-size: 10pt; line-height 12pt }
```

Marginaler och ramlinjer

margin styr hur bred marginalen runt objektet ska vara och anges med måttenheter, i procent eller som **auto**. Anges ett värde gäller det för alla sidor, anges fyra värden tolkas de i ordningen uppåt, höger, nedåt, vänster. Anges två eller tre värden tas de saknade värdena från motstående sidor. Marginalerna på de olika sidorna kan även anges individuellt med **margin-top**, **margin-right**, **margin-bottom** och **margin-left**.

padding styr fyllningen runt objektet och anges med måttenheter, i procent eller som **auto**. Anges ett värde gäller det för alla sidor, anges fyra värden tolkas de i ordningen uppåt, höger, nedåt, vänster. Anges två eller tre värden tas de saknade värdena från motstående sidor. Fyllningen på de olika sidorna kan även anges individuellt med **padding-top**, **padding-right**, **padding-bottom** och **padding-left**.

border-width kan anges som **thin**, **medium**, **thick** eller med måttenheter och styr bredden på en ramlinje runt objektet. Anges ett värde gäller det för alla sidor, anges fyra värden tolkas de i ordningen uppåt, höger, nedåt, vänster. Anges två eller tre värden tas de saknade värdena från motstående sidor. Ramlinjen på de olika sidorna kan även anges individuellt med **border-top-width**, **border-right-width**, **border-bottom-width** och **border-left-width**.

border-color anger färgen på en ramlinje runt objektet. Färgen anges som för color ovan. Anges ett värde gäller det för alla sidor, anges fyra värden tolkas de i ordningen uppåt, höger, nedåt, vänster. Anges två eller tre värden tas de saknade värdena från motstående sidor.

border-style anger typen av ramlinje och kan vara **none**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset** eller **outset**. Anges ett värde gäller det för alla sidor, anges fyra värden tolkas de i ordningen uppåt, höger, nedåt, vänster. Anges två eller tre värden tas de saknade värdena från motstående sidor. Webbläsare kan välja att visa alla ramlinjer som solid.

border kan användas för att styra border-width, border-style och border-color på samma gång. Värdena gäller för ramlinjens alla sidor. Vill man ange olika värden för de olika sidorna får man använda **border-top**, **border-right**, **border-bottom** och **border-left**.

```
BLOCKQUOTE { border: thin solid red }
```

width och **height** styr ett objekts bredd och höjd. Det är mest användbart för bilder, men kan i vissa webbläsare även användas för text. Värden kan anges med måttenheter, i procent eller som **auto**. Om det ena värdet anges med mått eller i procent, och det andra är auto storleksförändras bilden proportionerligt.

```
IMG { width: 125px; height: 50px }
IMG { width: auto; height: 20mm }
IMG { width: 200%; height: auto }
```

float styr hur text figursätts runt ett objekt. Det är mest användbart för bilder, men kan även användas för text. Möjliga värden är **left**, **right** och **none**. Left och right placerar objektet i respektive marginal. Den omgivande texten flyter runt om. None gör att objektet placeras i den löpande texten.

```
IMG.knapp { float: left }
```

clear anger ifall ett element tillåter att det finns flytande objekt vid dess sidor (se float ovan). Möjliga värden är **none**, **left**, **right** och **both**. Om man till exempel anger { clear: left } för ett element så kommer detta att flyttas ner under eventuella flytande objekt på vänster sida.

```
H1 { clear: left }
```

Klassifikation

display anger hur ett element hanteras och hur det visas. Möjliga värden är **block** (elementet startar på en ny rad), **inline** (elementet ligger på samma rad som omgivande element), **list-item** (elementet har en listsymbol framför sig) eller **none** (objektet och dess eventuella ramar etc visas inte alls). Webbläsare kan välja att ignorera denna kod.

white-space styr ifall flera mellanslag efter varandra ska slås ihop till ett (**normal**), visas upp som de står i koden (**pre**) eller ifall rader endast bryts med
-koder (**nowrap**). Webbläsare kan välja att ignorera denna kod.

list-style-type styr hur punkterna i en numrerad eller onumrerad lista ska se ut. Möjliga värden är **disc**, **circle**, **square**, **decimal**, **lower-roman**, **upper-roman**, **lower-alpha**, **upper-alpha** eller **none**. Se avsnittet om listor för exempel.

```
UL { list-style-type: lower-roman }
```

list-style-image anger att en bild ska användas som listpunkt. Kan vara en url eller **none**.

```
UL { list-style-image: url(http://www.firma.se/funny-star.gif) }
```

list-style-position anger ifall listpunkten ska ligga utanför listan (**outside**) som vanligt eller ligga indragen i listan (**inside**).

list-style kan användas för att styra list-style-type, list-style-image och list-style-position på samma gång.

```
UL { list-style: url(http://www.firma.se/funny-star.gif) outside }
UL UL { list-style: disc outside }   /* gäller för listor i listor (UL UL) */
OL { list-style: upper-alpha inside }
UL { list-style: url(smiley.gif) disc }   /* disc visas ifall bilden inte finns */
```

Måttenheter

Måttangivelser kan anges med följande olika enheter:

em det typografiska måttet fyrkant, d.v.s. lika långt som texten är hög. Indrag i början av stycken räknas ofta i em.

ex x-höjd, d.v.s. höjden av bokstaven x i aktuellt typsnitt.

px pixlar, d.v.s. bildpunkter på skärmen. Om storleken på en pixel skiljer sig väldigt mycket från vanliga skärmupplösningar, t.ex. vid utskrift, kommer detta värde att förändras proportionerligt.

cm centimeter.

mm millimeter.

in tum, d.v.s. 25,4 mm.
pt typografiska punkter enligt PostScript-systemet, d.v.s 1/72 tum.
pc pica enligt PostScript-systemet, d.v.s 12pt

URL:er

Adresser till bildfiler etc anges med koden **url** följt av adressen inom parentes (se exemplen ovan). Adressen kan, men behöver inte, omges med enkla (') eller dubbla (") citattecken.

Observera att relativa adresser tolkas utifrån den adress som style sheet-filen har, inte ifrån adressen till det dokument som eventuellt hämtar in style sheet-filen med <LINK>. Om stilanvisningarna anges i samma fil (i en <STYLE>-kod eller direkt i en enskild kod) blir detta samma sak.

Läs mer om CSS på http://www.w3.org/pub/WWW/TR/WD-css1

Imagemaps

Ett mer avancerat sätt att använda en bild som länk är en s.k. imagemap. En imagemap är en bild där olika områden i bilden är länkar till olika ställen. De aktiva ytorna kan vara rektanglar, cirklar eller oregelbundna månghörningar. Det finns två sätt att göra imagemaps på, server-side imagemaps eller client-side imagemaps.

Server-side imagemaps är den äldre varianten och fungerar så att uppgiften om var i bilden som användaren klickar skickas till ett speciellt imagemap-program på servern som omdirigerar webbläsaren till rätt sida. Server-side imagemaps kräver att du hämtar webbsidan från en server (det går alltså inte att testa sidan lokalt från din hårddisk) som har ett imagemap-program installerat (det brukar ingå i de flesta webbservrar).

Client-side imagemaps är en nyare metod som fungerar så att informationen om vilka ytor i bilden som är länkar till vilka sidor finns inbakad i HTML-filen och att det är webbläsaren som direkt avgör vilken sida som ska hämtas. Fördelarna med detta är att man slipper köra ett imagemap-program på servern, kommunikationen mellan server och klient blir lite effektivare och användaren ser dessutom direkt på statusraden vart en länk kommer leda (vid server-side imagemaps ser man bara de bild-koordinater som kommer att skickas till servern). Client-side imagemaps kan man också testa lokalt, eftersom de inte kräver att en server är inblandad.

Client-side imagemaps är modernare och smidigare och rekommenderas i de flesta fall. Netscape stöder client-side imagemaps från och med version 2.0.

Client-side imagemaps

När man gör en client-side imagemap lägger man in informationen om vilka ytor som ska vara länkar och vart de leder i HTML-dokumentet. Oftast lägger man den i samma HTML-dokument som bildreferensen. En del webbläsare tillåter också att man har den i ett godtyckligt annat dokument, t.ex. om samma imagemap ska finnas på flera webbsidor.

För att ange att en bild är en client-side imagemap använder man attributet **USEMAP:**

```
<IMG SRC="bilden.gif" USEMAP="#karta">
```

Mappningsinformationen anges med styrkoderna **<MAP>** och **<AREA>**:

```
<MAP NAME="karta">
<AREA SHAPE="rect" COORDS="10, 10, 40, 50" HREF="url">
</MAP>
```

Varje <MAP> tilldelas ett namn, i det här fallet ”karta”. Man anropar <MAP> från USEMAP med detta namn. Det får inte finnas flera <MAP> med samma namn i samma fil. Om anropet görs från en annan fil anges även filnamnet:

```
<IMG SRC="bilden.gif" USEMAP="mapfil.html#karta">
```

USEMAP pekar på platsen där mappningsinformationen finns på samma sätt som en hypertextlänk inom ett dokument. Tänk på syntaxen <A HREF="#namn"> för att göra länken och <A NAME="namn"> för att markera målpunkten. På samma sätt pekar USEMAP på mappningsinformationen.

Varje yta i bilden som ska fungera som en länk skrivs som en **<AREA>**-kod mellan <MAP> och </MAP>. Till varje <AREA>-kod anger man attributen **SHAPE** (som specifierar vilken form ytan ska ha), **COORDS** (som specifierar de koordinater som ytan har), **HREF** (som specifierar den webbadress som länken ska leda till), **NOHREF** (som anges om ytan inte ska fungera som länk) och **ALT** (som åstadkommer en alternativ text för textbaserade webbläsare).

Möjliga värden för SHAPE är **rect** (den aktiva ytan blir en rektangel), **poly** (den aktiva ytan blir en månghörning), **circle** (den aktiva ytan blir en cirkel) eller **default** (anger att den yta som ligger utanför de andra, definierade ytorna ska vara aktiv). Om man inte anger SHAPE antas SHAPE="rect".

Koordinaterna anges med **COORDS="a, b, c, d, ..."**. Värdena räknas i pixlar och koordinaterna (0,0) ligger i bildens övre vänstra hörn. För en rektangel anges koordinaterna som ”vänster, överkant, höger, underkant” (ett tal var). För en cirkel anges koordinaterna som mittpunktens x- och y-värde (två tal) samt radiens längd (ett tal). För en månghörning anges koordinaterna som parvisa x- och y-värden för varje av figurens hörn (x1, y1, x2, y2, ...).

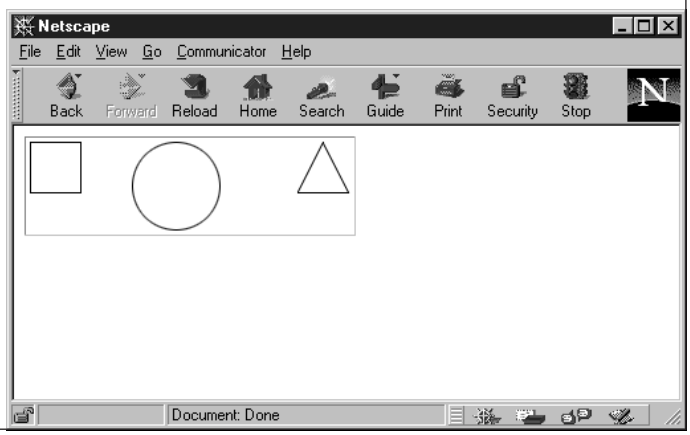
Adressen till det länkade dokumentet anges med **HREF="url"** på samma sätt som i <A HREF>. Tänk på att en relativ adress kommer att tolkas relativt den fil som mappningsinformationen ligger i, inte den där själva bildreferensen finns. Om <IMG USEMAP> och <MAP> <AREA> </MAP> ligger i samma fil, eller i olika filer i samma mapp, blir detta samma sak.

Om den specificerade ytan inte ska fungera som länk skriver man **NOHREF** istället för **HREF="url"**.

Det går att definiera hur många <AREA>-koder som helst. Om två ytor överlappar varandra kommer den som kommer först i listan gälla. En yta som inte definierats antas vara NOHREF, d.v.s. kommer inte fungera som länk.

```
<IMG SRC=”bilden.gif” USEMAP=”#karta”>

<MAP NAME="karta">
<AREA SHAPE="rect" COORDS="4,4,42,42"
  HREF="stork.html" ALT="Allt om storkarnas hemliga liv">
<AREA SHAPE="circle" COORDS="114,37,34" NOHREF>
<AREA SHAPE="poly" COORDS="225,4,244,42,206,42"
  HREF="mailto:info@firma.se" ALT="Skicka oss ett brev!">
<AREA SHAPE="default"
  HREF="struts.html" ALT="En novell om en struts">
</MAP>
```



I en imagemap fungerar olika ytor i en bild som länkar till olika webbsidor. Användaren kommer till olika ställen beroende på var i bilden hon klickar.

<AREA>-koderna kan man antingen skapa i en speciell map-editor (som till exempel Map This! för PC eller WebMap för Macintosh) eller skriva för hand. Vissa HTML-editorer har även denna funktion inbyggd. Om du skriver <AREA>-koderna för hand kan du hitta koordinaterna för de länkade ytorna genom att först skriva länken som en server-side imagemap (se nedan), titta på webbsidan och notera de koordinatpar som visas i Netscapes statusrad (längst ner i fönstret).

Server-side imagemaps

En server-side imagemap består av en bild, en länk, en fristående .map-fil och ett imagemap-program på servern. HTML-koden ser ut såhär:

```
<A HREF="/cgi-bin/imagemap/kaka/allan.map">
<IMG SRC="allan.gif" ISMAP BORDER="0"> </A>
```

<IMG SRC> visar vad bildfilen heter, och som attribut till IMG anges **ISMAP**. Eftersom bilden fungerar som länk så får man en blå ram runt den. Vill man inte ha en blå ram anges **BORDER="0"**.

Hur .map-filen och länken i <A HREF>-koden skrivs beror på imagemap-programmet och vilken server som används. Det finns två vanliga sätt att göra det på; NCSA-style och CERN-style.

Läs mer om NCSA-style imagemaps på http://hoohoo.ncsa.uiuc.edu/docs/tutorials/imagemapping.html

Läs mer om CERN-style imagemaps på http://www.w3.org/pub/WWW/Daemon/User/CGI/HTImageDoc.html

Både och

Det går att ange en bild både som client-side imagemap och server-side imagemap på samma gång:

```
<A HREF="/cgi-bin/imagemap/pic2.map">
<IMG SRC="/images/tech/pic2.gif" USEMAP="maps.html#map2" ISMAP>
</A>
```

Då anger man mappningsinformationen både i en .map-fil och med styrkoderna <MAP> och <AREA>. Webbläsare som stöder client-side imagemaps kommer att använda informationen inom <MAP> och <AREA>, medan de som inte stöder client-side imagemaps kommer att använda .map-filen på servern.

Metainformation

Styrkoden **<META>**, som ska ligga i dokumentets ”head”-del, används för s.k. metainformation, d.v.s. information om själva dokumentet, till exempel nyckelord, författarens namn, datum för senaste uppdatering etc. Metainformationen visas inte på sidan, utan används av sökmaskiner och annat för att identifiera, indexera och katalogisera dokumentet. Metainformationen bör följa vissa regler för att underlätta sökning. Olika sökmaskiner följer olika regler. Se sökmaskinernas informationssidor för detaljer.

Du kan läsa mer om <META> i den fullständiga tekniska specifikationen för HTML som finns på World Wide Web Consortiums webbsidor på http://www.w3.org/

<META> används även vid ”client pull” (se nedan).

Client Pull och Server Push

Client Pull innebär att webbläsaren laddar om den nuvarande sidan med ett visst intervall eller hämtar en viss sida efter en viss tid. Detta görs med hjälp av <META HTTP-EQUIV>.

Exempel:

<META HTTP-EQUIV="Refresh" CONTENT="3"> Sidan laddas om var 3:e sekund.

<META HTTP-EQUIV="Refresh" CONTENT="5; URL=webbadress"> Webbklienten hoppar till den angivna webbadressen efter 5 sekunder. Webbadressen ska anges som en absolut adress (http://www.firma.se/fil.html). Notera hur citattecknen sitter.

Om den sida som webbläsaren hoppar till efter den angivna tiden också har en <META HTTP-EQUIV> som pekar någonstans kommer användaren kastas dit efter den utsatta tiden. Till exempel kan man på detta sätt göra ”bildspel” genom att omdirigera webbläsaren till nästa och nästa och nästa sida. Man kan också dirigera tillbaka webbläsaren till den första sidan, vilket kommer göra att webbläsaren växlar mellan de två sidorna med angivet intervall.

Observera att <META HTTP-EQUIV>-koden ska ingå i <HEAD> och alltså komma före texten på själva sidan.

Var väldigt försiktig med att använda Client Pull. Det är många användare som inte gillar att webbdesignern tvingar dem att byta sida. Om du ändå använder det, var noga med att erbjuda en länk bort från sidan. Annars kan det vara svårt att ”ta sig ur” eftersom ”Back”-knappen i

webbläsaren bara tar användaren tillbaka till förra sidan, som alltså direkt leder tillbaka till den nuvarande.

Ett annat sätt att uppdatera ett dokument kontinuerligt är Server Push. Normalt sett när man hämtar en sida kopplas en förbindelse upp, sidan skickas över och förbindelsen kopplas ner. Vid Server Push hålls förbindelsen öppen så att servern kan fortsätta att skicka data. Server Push användes tidigare framför allt för animationer men hann aldrig bli så populärt, förrän den betydligt enklare metoden med animerade GIF-filer slog igenom. Andra tänkbara användningsområden är webbkameror, löpande uppdatering av börskurser etc. Mer information om Server Push finns på http://home.netscape.com/assist/net_sites/pushpull.html

---

Frames

Med Frames kan du dela upp webbläsarens fönster i flera mindre rutor. I varje sådan ruta kan man ladda in varsin webbsida oberoende av de andra rutorna på sidan. Varje ruta kan uppdateras för sig, och varje ruta får egna rullningslister ifall det behövs.

Ett typiskt användningsområde för Frames är att ha en innehållsförteckning i en separat ruta bredvid själva webbsidan. På så vis finns innehållsförteckningen alltid tillgänglig och behöver inte laddas om varje gång användaren byter sida.

Frames fungerar inte i alla webbläsare, och därför bör man vara försiktig med att använda den här funktionen. De användare som har webbläsare som inte stöder Frames kommer att få se en helt tom sida, såvida inte webbdesignern har lagt in ett alternativt innehåll speciellt för dem.

Det finns en annan orsak till att vara restriktiv med användningen av Frames: många användare, även bland dem som har webbläsare som klarar Frames, är irriterade på funktionen och undviker sidor som är designade med Frames. Därför bör du bara använda Frames om detta verkligen tillför något av värde till din sida.

Frameset

Sidans uppdelning i rutor definieras med diverse attribut till styrkoden **<FRAMESET>**. Koden avslutas med **</FRAMESET>** och mellan dessa koder beskriver man vad de olika rutorna på sidan ska innehålla med hjälp av styrkoden **<FRAME>** (se nedan).

I varje ruta på sidan laddas en komplett webbsida in. Varje sådan sida är alltså ett eget HTML-dokument.

Varken styrkoderna <BODY> och </BODY> eller någon annan styrkod eller text som normalt sett placeras inom dessa koder får finnas före <FRAMESET>, annars kommer hela uppdelningen i Frames att ignoreras av webbläsaren. <FRAMESET> måste alltså komma direkt efter <HEAD> </HEAD>.

Rutuppdelning i sidled

Sidan delas upp i sidled med <FRAMESET **COLS="värdelista"**>. I värdelistan anges uppdelningen i pixlar, i procent eller som relativa värden. De olika värdena i värdelistan separeras med kommatecken.

Fast värde. Ett värde som anges som bara ett tal tolkas som rutans bredd i pixlar. Detta är användbart om rutan ska innehålla en bild med känd storlek, men kan vara vanskligt i andra sammanhang eftersom man inte vet hur stor en pixel är på användarens skärm. Om man använder fasta värden bör man blanda dem med procentvärden eller relativa värden så att alla rutor tillsammans fyller 100% av användarens fönster.

Procentvärde. Ett tal med ett procenttecken direkt efter tolkas som rutans bredd i procent av hela fönstrets bredd. Om summan av alla procentvärden är större än 100 kommer webbläsaren att minska alla värden proportionerligt. Om summan är mindre än 100 och det finns relativa värden kommer de att få den extra platsen. Om det inte finns några relativa värden kommer alla procentvärden ökas proportionerligt.

Relativa värden. Ett ensamt *-tecken ger en ruta som är så bred som det går, efter att rutor med fast värde och procentvärde fått plats. Om det finns flera relativa värden (d.v.s. flera ensamma *-tecken) kommer den tillgängliga platsen fördelas lika mellan dem. Om det finns en siffra framför *-tecknet kommer den rutan få motsvarande utrymme. "2*,*" ger den första rutan 2/3 av utrymmet, och 1/3 till den andra.

Om COLS inte anges alls tolkas detta som att fönstret inte är uppdelat på bredden. Exempel:

<FRAMESET COLS="50%,50%"> Fönstret delas på bredden i två lika stora rutor.

<FRAMESET COLS="10%,80%,10%"> En smal ruta på vardera sidan och en bred i mitten.

<FRAMESET COLS="200,*"> En 200 pixlar bred ruta till vänster, och en så bred som fönstret tillåter till höger.

Rutuppdelning i höjdled

Sidan delas upp i höjdled med <FRAMESET **ROWS="värdelista"**>. ROWS fungerar på samma sätt som COLS. Ofta börjar man med att dela upp fönstret på bredden med ett yttre <FRAMESET> och sedan delar man upp varje sådan ruta på höjden med flera underordnade <FRAMESET>.

Man kan också göra tvärt om; dela in på höjden först och på bredden sedan. Det är också tillåtet att dela upp på bredden och höjden samtidigt i samma <FRAMESET> – detta åstadkommer ett rutnät.

Definiera rutornas innehåll och egenskaper

I varje ruta på sidan laddas en komplett webbsida in. Vilken sida som ska laddas in anges, tillsammans med några andra egenskaper, med styrkoden **<FRAME>**. Man bestämmer vad rutan ska innehålla, vad den ska heta, om den ska få ha rullningslister, om användaren ska få ändra storlek på den och hur breda marginaler den ska ha. <FRAME> har ingen slutkod. Till <FRAME> kan man lägga följande attribut:

SRC="url" anger webbadressen till det dokument som ska laddas in i rutan. Den anges som vanligt antingen som relativ adress eller absolut adress. Rutor som saknar SRC kommer att visas som blanka rutor i den storlek som angetts i <FRAMESET>.

NAME="rutans_namn" används för att ge en ruta ett namn så att den kan bli refererad till från länkar i andra rutor inom samma fönster. Man behöver inte ange NAME, då blir rutan namnlös. Om man vill att länkar i en ruta ska göra så att webbsidor laddas in i andra rutor måste man namnge dessa. Namngivna rutor kan adresseras med TARGET (se nedan). Ett namn måste börja med en bokstav eller siffra och får bestå av bokstäver a-z, siffror, bindstreck och understreck.

Med **MARGINWIDTH="value"** och **MARGINHEIGHT="value"** kan man ange extra marginal mellan rutans kant och innehållet i rutan. MARGINWIDTH anger marginalen åt vänster och höger medan MARGINHEIGHT anger marginalen uppåt och nedåt. Värdet räknas i pixlar. Värdet kan inte vara mindre än ett. Anges inte MARGINWIDTH och MARGINHEIGHT kommer webbläsaren själv att avgöra vilken marginal som behövs.

Normalt sett drar webbläsaren en kantlinje mellan de olika rutorna i fönstret. Med **FRAMEBORDER="0"** anger man att rutan inte ska ha en kant. Observera att det ändå kan bli en kantlinje, ifall angränsande ruta har kantlinje specifierad, antingen genom att ha FRAMEBORDER="1" eller genom att sakna FRAMEBORDER-attributet.

SCROLLING="yes|no|auto" anger ifall rutan ska ha en rullningslist eller ej. Anges ”yes” kommer rutan alltid ha en rullningslist, anges ”no” kommer den aldrig ha det, och anges ”auto” kommer webbläsaren sätta in rullningslister när det behövs. Anges inte SCROLLING alls kommer webbläsaren även då att sätta in rullningslister när det behövs.

Normalt sett kan användaren ändra storleken på en ruta genom att dra rutans kant till önskat läge. Om man anger **NORESIZE** i <FRAME>-koden så tillåts inte detta.

Kantlinje mellan rutorna

Förutom att man kan visa eller ta bort kantlinjen mellan rutorna genom att ange attributet FRAMEBORDER till styrkoden <FRAME> (enligt ovan) tillåter också en del webbläsare att man anger attributen **BORDER="0"** och/eller **FRAMEBORDER="0"** till styrkoden <FRAMESET>. Med dessa attribut, som inte stöds av alla webbläsare, kan man även ange valfri bredd på kantlinjen, till exempel med BORDER="5".

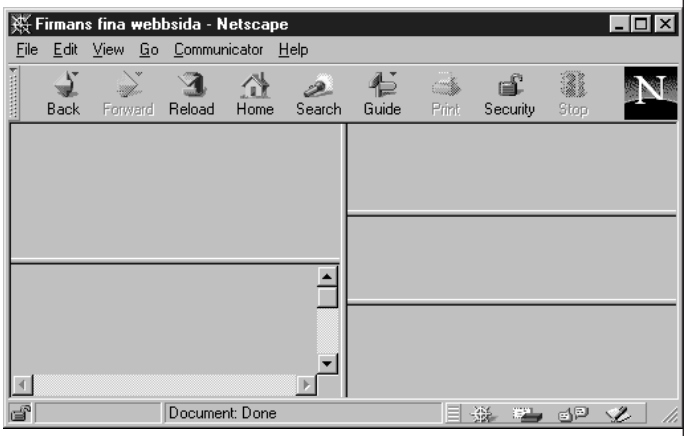
Alternativ sida

Webbläsare som inte stöder Frames kommer att visa upp en helt tom sida, såvida inte webbdesignern har lagt in ett alternativt innehåll speciellt för dem. Detta görs mellan styrkoderna **<NOFRAMES>** och **</NOFRAMES>**. En webbläsare som hanterar Frames ignorerar allt som ligger mellan dessa styrkoder.

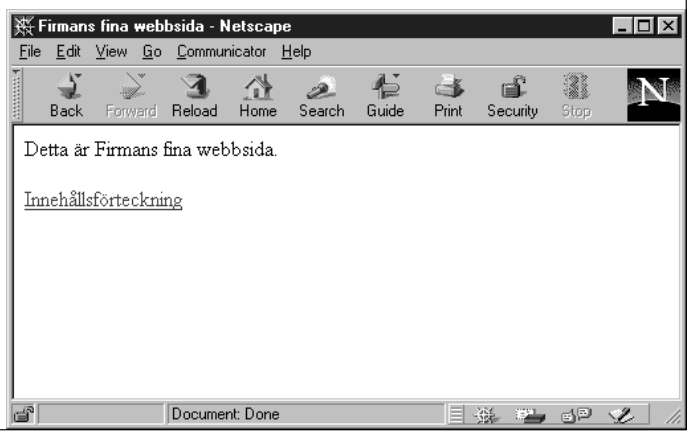
Var noga med att erbjuda ett vettigt alternativ här, speciellt på din webbplats framsida. Att bara skriva att besökaren har fel typ av webbläsare är oartigt. Om ditt frameset till exempel består av två rutor; en med en innehållsförteckning och en med själva sidorna så är en enkel noframes-lösning att erbjuda en länk till sidan med innehållsförteckningen. Om du har tillgång till server-side includes (se detta avsnitt) kan du rent av lägga in hela innehållsförteckningen som ”include file”.

Notera att man med <NOFRAMES> kan dölja innehåll för webbläsare som visar Frames, även på webbsidor som inte alls är indelade i rutor.

```
<HTML>
<HEAD><TITLE>Firmans fina webbsida</TITLE></HEAD>
<FRAMESET COLS=”50%,50%”>
  <FRAMESET ROWS=”50%,50%”>
    <FRAME SRC=”allan.html” NORESIZE NAME=”left-1”>
    <FRAME SRC=”kakan.html” SCROLLING=”yes” NAME=”left-2”>
  </FRAMESET>
  <FRAMESET ROWS=”33%,33%,33%”>
    <FRAME SRC=”innehall.html” NAME=”right-1”>
    <FRAME SRC=”stork.html” NAME=”right-2”>
    <FRAME SRC=”http://en.annan.server.se/” NAME=”right-3”>
  </FRAMESET>
</FRAMESET>
<NOFRAMES>
  <BODY BGCOLOR="#FFFFFF">
    Detta är Firmans fina webbsida.<P>
    <A HREF=”innehall.html”>Innehållsförteckning</A>
  </BODY>
</NOFRAMES>
</HTML>
```



Webbläsarens fönster delas upp i fem rutor som fylls med var sin separat webbsida.



Om besökaren använder en webbläsare som inte stöder Frames, eller en där användaren kan välja att slå av Frames-funktionen, kommer hon att se den alternativa sidan.

Styra länkar till rätt ruta

Normalt sett när en användare klickar på en länk kommer det nya dokumentet att visas i samma fönster som det förra. Detsamma gäller på en sida uppdelad i rutor. Varje ruta kan ses som ett separat fönster, och det som händer i en ruta eller ett fönster påverkar inte innehållet i de andra. Men ofta vill man att en länk i en ruta (till exempel i en innehållsförteckning) ska göra att den aktuella webbsidan laddas in i en annan ruta på sidan.

Med attributen **NAME** och **TARGET** kan webbdesignern namnge de olika rutorna och sedan tvinga webbsidor att hamna i dessa. En rutas namn är inte det samma som fönstertiteln som anges med <TITLE>.

En rutas namn definieras med **<FRAME NAME="rutans_namn">** (se ovan) och för att tvinga ett länkat dokument att hamna i just den rutan anger man TARGET i själva länken:

<A HREF="url" TARGET="rutans_namn">Allan tar kakan</A>

När användaren klickar på texten Allan tar kakan kommer dokumentet med adressen ”url” laddas in i rutan med namnet ”rutans_namn”.

Om alla (eller de flesta) länkar i en ruta ska ladda in dokument i en viss, annan ruta kan man ange **<BASE TARGET="rutans_namn">**

Anger man ett TARGET-attribut i en individuell <A>-kod har den företräde framför TARGET-attributet i <BASE>. <BASE> ska ligga i ”head”-delen av dokumentet.

TARGET fungerar även i en client-side imagemap. Exempel: <AREA SHAPE="rect" COORDS="12, 10, 25, 30" HREF="url" TARGET="rutans_namn">

TARGET kan också sättas i styrkoden <FORM>. Detta gör att resultatet från ett formulär i en ruta (det som servern skickar tillbaka till användaren efter att hon klickat på ”submit”) hamnar i en annan ruta. Exempel: <FORM ACTION="url" TARGET="rutans_namn">

Om man anger en TARGET-länk till ett rute-NAME som inte finns kommer istället ett nytt fönster att öppnas som då tilldelas det namnet.

Magiska namn

Namnet på en ruta måste börja med en bokstav eller en siffra för att vara giltigt. Alla andra namn kommer att ignoreras. Undantaget är fyra ”magiska” namn som alla börjar med ett understrykningstecken.

TARGET="_blank" Det länkade dokumentet kommer att laddas in i ett nytt, tomt fönster utan namn.

TARGET="_self" Det länkade dokumentet kommer att laddas in i samma ruta eller fönster som länken finns i. Detta är användbart för att gå förbi ett rutenamn som satts i <BASE>.

TARGET="_parent" Det länkade dokumentet kommer att laddas in i närmaste överordnat FRAMESET. Finns det inget sådant laddas dokumentet in i samma ruta eller fönster som länken finns i (samma som för _self).

TARGET="_top" Det länkade dokumentet kommer att laddas in i hela det aktuella fönstret. Alla indelningar i rutor kommer att försvinna. Detta är användbart för att komma ur många nivåer av ruteindelningar (d.v.s. en ruta som i sin tur är uppdelad i rutor). Om sidan inte innehåller några rutor blir effekten samma som för _self.

Den vanligaste användningen av TARGET är för att tvinga länkade dokument att visas i rätt ruta. Men TARGET kan också användas utan rutor. Effekten blir då att dokumenten hamnar i olika fönster. Detta kan kanske vara användbart ibland, exempelvis för att behålla besökare kvar på sina sidor, även när man erbjuder länkar bort från dem, men för det allra mesta är det bara irriterande. Användarens skärm fylls av fönster med olika webbsidor i, istället för att visa de nya webbsidorna i samma fönster hela tiden.

Behörighetskontroll

Ibland gör man sidor som bara vissa, utvalda personer ska få se. Det enklaste sättet att ordna det är att bara berätta adressen till sidorna för dem som man vill ska kunna se dem, och heller inte ha några länkar till dem. Tänk bara på att någon annan kan länka till dina sidor utan att du vet om det. Detta kommer inte hindra dem som ”råkar” få reda på adressen.

Vill man ha ett mer effektivt skydd kan man kräva lösenord av användaren när hon försöker klicka fram sidan, eller bara visa sidan för dem som kopplar upp sig från vissa nät, domäner eller IP-nummer. Det går också att kombinera dessa metoder.

User Authentication (lösenord)

En effektiv form av behörighetskontroll är användarnamn+lösenord. Detta är en funktion som ligger i webbservern. Man kan lösenordsskydda vissa mappar på servern. Alla filer som ligger i en och samma mapp kommer då att omfattas av samma lösenord. En del servrar tillåter även att man lösenordsskyddar enstaka filer. Användaren kommer att få ange användarnamn och lösenord första gången hon tittar på en sida i den aktuella mappen under den sessionen (d.v.s. webbläsaren minns lösenordet tills man stänger av den). Man kan välja en uppsättning användarnamn+lösenord som alla använder, eller ha individuella uppsättningar för varje person.

Class Restrictions (spärra vissa IP-nummer eller domäner)

Alla datorer kopplade till Internet har ett eget unikt nummer, ett IP-nummer. Många har även ett eller flera namn, grupperade i domäner. Man kan basera åtkomstkontrollen på detta, till exempel bara godkänna uppkopplingsförsök från IP-nummer som börjar på 193.44.96.x eller maskinnamn som slutar på .firma.se.

Ställa in servern

Hur man gör behörighetskontrollen beror på vilket serverprogram man använder. En metod som bland annat servrarna NCSA och Apache använder är att man lägger en speciell fil kallad **.htaccess** i den mapp man vill skydda. Filen innehåller information om vilka åtkomstrestriktioner som gäller för filerna i mappen, till exempel vilka användarnamn eller vilka IP-nummer som tillåts se sidorna.

De lösenord som ska användas sparas i krypterad form i en annan fil som läggs i en annan mapp, helst en som inte går att nå via webben (d.v.s. ej i document root). Mer information om hur .htaccess-filen ska se ut finns på http://hoohoo.ncsa.uiuc.edu/docs/tutorials/user.html

En del serverprogram, speciellt sådana som körs under Windows NT, kan inte fjärrstyras med .htaccess-filer utan där sköts åtkomstrestriktionerna från webbserverprogrammets inställningspanel, d.v.s. man måste vara vid själva maskinen för att ändra något.

Spärra trafik i router/brandvägg

Förutom att hindra uppkopplingsförsök i servern kan man också spärra HTTP-trafik i routern eller brandväggen som kopplar ihop firmans lokala nät med stora Internet. Spärren kan vara fullständig (d.v.s. bara anrop från samma lokala nät tillåts) eller beroende på vilket annat nät eller IP-nummer som anropet kommer ifrån. Detta kan endast den systemadministratör som ställer in routern eller brandväggen göra.

Formulär

Hittills har vi sparat information på servern, som har skickat den till användaren på begäran. Men ibland vill man att användaren ska skicka information tillbaka till servern. Det kan handla om att fylla i uppgifter av olika slag, kryssa i rutor i enkäter, beställa varor, ange sökord till en databassökning etc. Alltid när användaren ska avge information från en webbsida använder man formulär. Informationen som användaren skickar in kan användas till en automatisk beräkning eller databehandling, sparas på servern, eller skickas via e-post för manuell behandling. Denna databehandling görs av ett s.k. CGI-program som är installerat på serverdatorn.

Själva formuläret kodas i HTML med speciella styrkoder. CGI-programmet skrivs i ett lämpligt programmeringsspråk och installeras och körs (exekveras) på servern. På de flesta webbservrar finns det några färdiga CGI-program installerade, och det finns också färdiga CGI-program att hämta på Internet. Se vidare avsnittet om CGI.

Ett formulär börjar alltid med **<FORM>** och slutar med **</FORM>**. Däremellan kan man placera textinskrivningsfält, radioknappar, kryssrutor, popup-menyer etc. Formuläret kan också innehålla ”vanliga” HTML-konstruktioner som text, rubriker, listor, tabeller etc. Man får dock inte ha ett formulär inne i ett annat formulär (men man kan ha två eller flera formulär efter varandra på samma sida).

Till styrkoden <FORM> anger man attribut som anger vart och hur formulärdatat (den information som användaren matat in) ska skickas.

Med attributet **METHOD** anger man hur formulärdatat ska skickas. Man kan välja mellan antingen **GET** eller **POST**.

GET innebär att formulärdatat skickas som en del av URL:en, efter ett frågetecken. Formulärdatat kommer att synas i webbläsarens adressfält. Mängden data som kan skickas på detta sätt är begränsad.

POST innebär att formulärdatat skickas som ett datablock till servern tillsammans med begäran om den nya sidan. Denna metod är att föredra i de allra flesta fall.

Med attributet **ACTION** anger man vart formulärdatat ska skickas. Detta är oftast webbadressen (URL:en) till ett CGI-program på servern (se vidare avsnittet om CGI). CGI-programmet tar hand om formulärdatat, behandlar det på något sätt och skickar sedan tillbaka en webbsida till webbläsaren, baserad på den information som användaren skickade in. Om formuläret till exempel används för att skriva in sökbegrepp till en databassökning så kan CGI-programmet skicka tillbaka en sida med sökresultatet på.

Exempel: <FORM METHOD="POST" ACTION="/cgi-bin/dbsearch"> gör att formulärdatat skickas med metoden POST till CGI-programmet ”dbsearch” i den egna serverns cgi-bin-mapp.

Man kan också ange ACTION som en mailto:-adress. Då skickas formulärdatat som ett e-postmeddelande till angiven adress. E-postmeddelandet går från webbläsaren, och för att detta ska fungera så måste användaren ha skrivit in sin e-postadress och sin server för utgående e-post i sin webbläsares inställningar.

Exempel: <FORM METHOD="POST" ACTION="mailto:info@firma.se"> gör att formulärdatat skickas med metoden POST till e-postadressen info@firma.se.

Om man låter ACTION leda till en e-postadress kan man också ange ENCTYPE="text/plain" för att det ska bli lättare för brevets mottagare att läsa formulärdatat.

NAME/VALUE-par

Det som användaren skrivit in skickas till servern parvis som NAME och VALUE. Varje textfält, kryssruta etc har ett namn (som specifieras av webbdesignern) och ett värde (som tilldelas av den användare som fyller i formuläret). Dessa NAME/VALUE-par skickas sedan till servern.

Textinskrivningsfält

Den enklaste typen av formulärfält görs med **<INPUT NAME="entry">**. Resultatet på webbsidan är ett 20 tecken långt, enradigt fält där användaren kan skriva in text. Om användaren skriver in mer än 20 tecken rullar texten åt vänster. ”entry” är ett godtyckligt namn på just detta fält. Har du flera textfält på sidan så ska dessa ha olika namn. Fältens namn syns inte på webbsidan. Välj gärna namn som har med fältets syfte att göra. Exempel:

```
Ditt namn: <INPUT NAME="namn"><P>
Ditt telefonnummer: <INPUT NAME="telefon"><P>
Din skostorlek: <INPUT NAME="sko"><P>
```

Vill du ha ett textinskrivningsfält längre eller kortare än 20 tecken använder du **SIZE**:

```
Din skostorlek: <INPUT NAME="sko" SIZE="3"> Textfältet blir tre tecken långt.
```

Observera att det fortfarande går att skriva in mer text, det som inte syns i rutan rullar åt vänster. Om du vill begränsa användarna att bara skriva in så mycket som får plats i rutan använder du **MAXLENGTH**:

```
Din skostorlek: <INPUT NAME="sko" SIZE="3" MAXLENGTH="3">
```

Textfältet blir tre tecken långt och försöker man skriva in mer hörs ett varningsljud. Värdena för SIZE och MAXLENGTH kan vara olika.

Det går att fylla i fältet i förväg. Genom att ange VALUE så kommer fältet ha text i sig när det presenteras för användaren. Användaren kan sedan välja att ändra texten eller inte. Detta kan till exempel användas för att föreslå ett svar eller för att visa hur ett svar kan se ut.

```
<INPUT NAME="namn" VALUE="Fyll i ditt namn här">
```

Sänd-knapp

När användaren fyllt i formuläret ska hon klicka på en knapp för att skicka iväg informationen till servern. Denna knapp görs med:

<INPUT **TYPE="submit"** VALUE="Skicka formuläret">

VALUE anger den text som står på själva knappen. Om denna inte anges kommer det stå ”Submit” på knappen.

Man kan ha flera sänd-knappar i samma formulär.

Återställnings-knapp

Bredvid sänd-knappen brukar man ibland sätta en återställnings-knapp (reset-knapp) som användaren kan klicka på ifall hon ångrar sig och vill tömma alla ifyllda fält och sedan börja om.

<INPUT **TYPE="reset"** VALUE="Töm alla fält">

VALUE är den text som står på själva knappen. Om denna inte anges kommer det stå ”Reset” på knappen.

Ifall något av fälten har förvald text (med VALUE) så kommer denna att återställas. Reset innebär alltså ”återställ till förvalda värden” snarare än ”töm alla fält”.

Kryssrutor

För att göra en ruta som användaren kan välja att antingen kryssa i eller inte kryssa i skriver du:

<INPUT **TYPE="checkbox"** NAME="allan" VALUE="kakan">

<INPUT TYPE="checkbox" NAME="allan" VALUE="kakan" CHECKED>

CHECKED gör att rutan är ikryssad som förval.

Flera olika kryssrutor kan ha samma NAME. För varje ikryssad ruta kommer ett NAME/VALUE-par att skickas till servern. För icke ikryssade rutor skickas inget NAME/VALUE-par.

Radioknappar

Radioknappar är ungefär som kryssrutor, användaren kan välja att antingen kryssa i dem eller inte. Skillnaden är att bara en i en grupp av flera radioknappar med samma NAME kan vara ikryssad samtidigt. Endast ett NAME/VALUE-par kommer att skickas till servern. Om ingen av radio-knapparna markerats skickas inte något NAME/VALUE-par alls.

I exemplet nedan måste man välja en betalningsmetod, man kan inte både betala med kontanter och check samtidigt. Alla knappar i ett formulär som har samma NAME hör ihop, och endast en av dessa kan vara markerad.

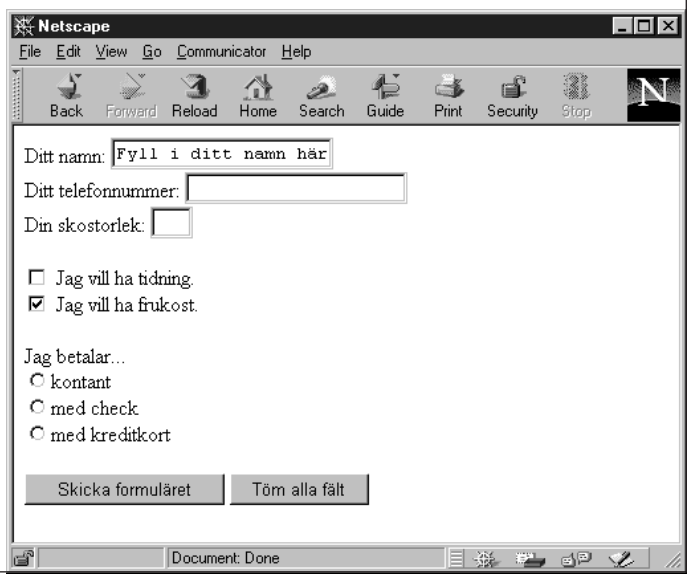
<INPUT **TYPE="radio"** NAME="betala" VALUE="kontant">Kontant.<P>

<INPUT TYPE="radio" NAME="betala" VALUE="check">Check.<P>

<INPUT TYPE="radio" NAME="betala" VALUE="kreditkort">Kreditkort.<P>

Om man vill att en av knapparna ska vara ikryssad som förval används attributet CHECKED (se ovan).

```
<FORM METHOD=”POST” ACTION=”/cgi-bin/dbsearch”>
Ditt namn: <INPUT NAME=”namn” VALUE=”Fyll i ditt namn här”><BR>
Ditt telefonnummer: <INPUT NAME=”telefon”><BR>
Din skostorlek: <INPUT NAME=”sko” SIZE=”3”>
<P>
<INPUT TYPE=”checkbox” NAME=”allan” VALUE=”kakan”>
Jag vill ha tidning.<BR>
<INPUT TYPE=”checkbox” NAME=”allan” VALUE=”kakan” CHECKED>
Jag vill ha frukost.
<P>
Jag betalar...<BR>
<INPUT TYPE=”radio” NAME=”betala” VALUE=”kontant”>kontant<BR>
<INPUT TYPE=”radio” NAME=”betala” VALUE=”check”>med check<BR>
<INPUT TYPE=”radio” NAME=”betala” VALUE=”kreditkort”>med kreditkort
<P>
<INPUT TYPE=”submit” VALUE=”Skicka formuläret”>
<INPUT TYPE=”reset” VALUE=”Töm alla fält”>
</FORM>
```



Utseendet på kryssrutor, radioknappar, send- och resetknappar varierar mycket mellan olika versioner av de populära webbläsarna.

Popupmeny

Ett annat sätt att låta användaren välja en av flera alternativ är popup-menyn. Den kan bli snyggare, speciellt om man har många olika saker att välja emellan. En popup-meny gör man med **<SELECT>** och **</SELECT>**. Mellan start- och slutkoderna lägger man in flera olika **<OPTION>**-koder. Exempel:

```
<SELECT NAME="mat">
<OPTION>Hamburgare
<OPTION>Pizza
<OPTION>Sushi
<OPTION SELECTED>Köttbullar
<OPTION>Kebab
<OPTION>Fylld baguette
</SELECT>
```

Det alternativ som står efter den första <OPTION>-koden kommer att bli det som syns ifall användaren inte ändrar något. Vill man att ett annat av alternativen ska vara förvalt anger man det som <OPTION **SELECTED**>. En reset-knapp återställer till förvalt värde.

Det NAME som skickas kommer att vara det som anges i styrkoden <SELECT>, och det VALUE som skickas är den text som står efter <OPTION>. Den syns även i listan på själva webbsidan. Ifall man vill att något annat än det som står i listan ska skickas till servern kan man ange **VALUE** inne i <OPTION>-koden:

```
<OPTION VALUE="Baguette">Fylld baguette
```

Om användaren i detta fall väljer ”Fylld baguette” från listan kommer NAME/VALUE-paret ”mat=Baguette” skickas till servern.

Rullningsbara listor med ett eller flera val

Ytterligare ett annat sätt att låta användaren välja en av flera alternativ är den rullningsbara listan. Den låter dessutom användaren välja flera av alternativen ifall webbdesignern så tillåter. En rullningsbar lista görs på samma sätt som popup-menyn. Enda skillnaden är att man anger hur mycket av listan (hur många rader) som ska synas med attributet **SIZE**. Är SIZE="2" eller större blir det en rullningslista, är SIZE="1" eller saknas blir det en popup-meny. Är listan längre än det antal rader som specificerats med SIZE får listan automatiskt en rullningslist.

```
<SELECT NAME="mat" SIZE="4">
<OPTION>Hamburgare
<OPTION>Pizza
<OPTION>Sushi
<OPTION SELECTED>Köttbullar
<OPTION>Kebab
<OPTION VALUE="Baguette">Fylld baguette
</SELECT>
```

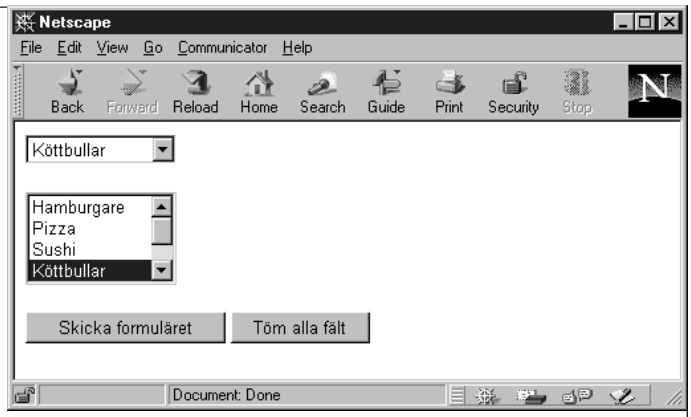
Även här kan man förvälja alternativ med <OPTION SELECTED> och ange speciella värden med <OPTION VALUE>. En reset-knapp återställer till förvalt värde.

I en rullningsbar lista kan användaren få markera flera alternativ. Detta görs möjligt genom attributet **MULTIPLE**:

```
<SELECT NAME="mat" MULTIPLE SIZE="4">
```

Här bör man skriva en anvisning på sidan så att användaren förstår att det är tillåtet att välja flera alternativ. Beroende på vad användaren har för datortyp markeras flera val med hjälp av skift-, control, kommando- eller motsvarande tangent.

```
<FORM METHOD=”POST”
 ACTION=”/cgi-bin/dbsearch”>
<SELECT NAME=”mat”>
<OPTION>Hamburgare
<OPTION>Pizza
<OPTION>Sushi
<OPTION SELECTED>Köttbullar
<OPTION>Kebab
<OPTION>Fylld baguette
</SELECT>
<P>
<SELECT NAME=”mat” SIZE=”4”>
<OPTION>Hamburgare
<OPTION>Pizza
<OPTION>Sushi
<OPTION SELECTED>Köttbullar
<OPTION>Kebab
<OPTION VALUE=”Baguette”>Fylld baguette
</SELECT>
<P>
<INPUT TYPE=”submit” VALUE=”Skicka formuläret”>
<INPUT TYPE=”reset” VALUE=”Töm alla fält”>
</FORM>
```



Textinskrivningsfält med flera rader

Vill man låta användaren skriva in flera rader text gör man detta med **<TEXTAREA>**. Denna styrkod kräver start- och slutkod.

<TEXTAREA NAME="brev_till_allan" ROWS="20" COLS="60"> </TEXTAREA>

Textfältet blir i detta fall 20 rader högt och 60 tecken brett. Skriver användaren in mer kommer fältet rullas.

Vill man att raderna i textinskrivningsfältet ska brytas när man når kanten på fältet istället för att texten bara rullas åt vänster kan man använda attributet **WRAP**. WRAP="SOFT" gör att raderna bryts på skärmen, men att webbläsaren inte stoppar in några radbrytningar i den text som skickas till servern, medan WRAP="HARD" gör att radbrytningarna även skickas med till servern. WRAP="OFF" ger samma effekt som att utelämna WRAP; texten ser ut och skickas exakt som den skrevs in. WRAP stöds endast av vissa webbläsare.

Vill man ha en text inskriven i förväg i ett <TEXTAREA>-fält anger man den mellan start- och slutkoderna:

<TEXTAREA NAME="brev_till_allan" ROWS="20" COLS="60">Hej Allan! Vill du ha en kaka?</TEXTAREA>

Användaren som fyller i formuläret kan ändra texten. En reset-knapp återställer till förvalt värde.

Lösenordsfält

När det som skrivs in i ett enkelradigt textinskrivningsfält ska vara hemligt för den som kanske står och tittar användaren över axeln (till exempel lösenord) kan man använda **TYPE="password"**:

<INPUT TYPE="password" NAME="pwd">

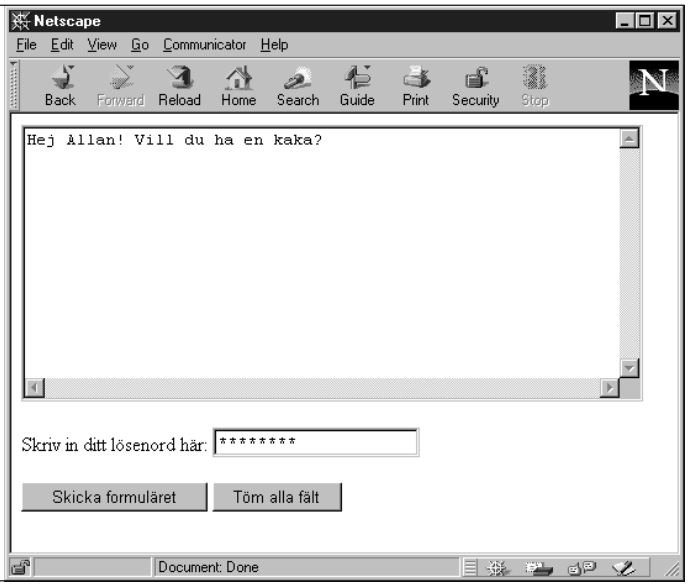
Det som användaren skriver in kommer visas som asterisker eller svarta cirklar.

Även här kan man ange ett förval t.ex. för anslutning som gäst:

<INPUT TYPE="password" NAME="pwd" VALUE="guestpwd">

Användaren som fyller i formuläret kan ändra texten. Notera att ett förvalt värde lätt kan ses av en användare som tittar på HTML-koden via ”View Document Source”. En reset-knapp återställer till förvalt värde.

```
<FORM METHOD=”POST”
 ACTION=”/cgi-bin/dbsearch”>
<TEXTAREA
 NAME=”brev_till_allan”
 ROWS=”12” COLS=”60”>Hej Allan!
 Vill du ha en kaka?</TEXTAREA>
<P>
Skriv in ditt lösenord här:
<INPUT TYPE=”password”
 NAME=”pwd” VALUE=”guestpwd”>
<P>
<INPUT TYPE=”submit”
 VALUE=”Skicka formuläret”>
<INPUT TYPE=”reset”
 VALUE=”Töm alla fält”>
</FORM>
```



Gömda fält

Ibland vill man skicka information till servern som användaren inte ska behöva se eller kunna ändra, till exempel om man har ett ganska generellt CGI-program som flera kan använda genom att styra det med gömda fält, eller för att ”ta med sig” information från ett formulär till ett annat. Ett gömt fält kan innehålla helt valfria NAME/VALUE-par och anges med **TYPE="hidden"**:

<INPUT TYPE="hidden" NAME="stork" VALUE="gris">

Observera att även gömda fält lätt kan ses av en användare som tittar på HTML-koden via ”View Document Source”.

Bilder

Med attributet **TYPE="image"** kan man låta olika saker hända beroende på var i en bild som användaren klickar. Till exempel kan man skriva ett CGI-program som utför en beräkning utifrån var i ett diagram eller en karta som användaren klickar.

<INPUT NAME="Allan" TYPE="image" SRC="http://www.firma.se/bild.gif">

På webbsidan visas bilden som anges som SRC, och när användaren klickar på bilden skickas ett NAME/VALUE-par där VALUE är de koordinater inom bilden som användaren klickade på.

Uppladdning av en fil

Med attributet **TYPE="file"** kan man låta användaren skicka in en fil till servern. Webbläsaren frågar på något sätt användaren efter ett filnamn, till exempel genom att lägga ut ett textfält där filnamnet ska skrivas in och/eller en ”Browse...”-knapp på webbsidan. Genom att trycka på ”Browse...”-knappen får användaren upp sitt datorsystems normala dialogruta för att öppna en fil.

Exakt hur detta ser ut beror helt på webbläsaren och operativsystemet. Exempel:

```
<FORM METHOD="POST" ACTION="url" ENCTYPE="multipart/form-data">
<INPUT NAME="filename" TYPE="file">
</FORM>
```

Filen skickas, tillsammans med övrig formulärdata, som mediatypen ”multipart”. CGI-programmet på servern som tar emot filen måste hantera detta på något lämpligt sätt.

CGI – Common Gateway Interface

Hittills har vi skrivit HTML-filer i en editor av något slag. HTML-filen har sparats på din egen hårddisk eller på en serverdators hårddisk. HTML-filen har sedan hämtats med en webbläsare, tolkats och visats upp i dess fönster. Tills vi går in och ändrar i HTML-filen ser webbsidan exakt likadan ut. Innehållet är statiskt.

Med CGI däremot kan man skapa dynamiska webbsidor. Innehållet på sidorna kan påverkas av olika saker som till exempel tiden på dagen, vem som tittar på sidan, vilken webbläsare som används, innehållet i en databas, eller beroende på den information som användaren fyllt i i ett formulär.

Tekniskt fungerar det så att istället för att servern skickar en färdig HTML-fil till klienten, så startar servern ett dataprogram, ett CGI-program. Programmet gör någon form av beräkning eller databehandling, och lämnar sedan ifrån sig utdata som servern skickar till klienten. En sida som skapats på detta sätt är vanlig HTML, men HTML-koden sparas aldrig som en fil på en hårddisk, utan skapas ”i farten” som utdata från ett CGI-program.

Indata till CGI-programmet kan bland annat tas från serverns systemklocka, filer på serverns hårddisk, från det som användaren har fyllt i i ett formulär eller från s.k. environment-variabler som innehåller information om uppkopplingen (se nedan).

CGI-programmet skrivs i något programmeringsspråk som går att köra på den dator som webbservern går på. Ett av de vanligaste programmeringsspråken för att skriva CGI-program är Perl, dels för att Perl går att köra på en mängd olika datorer utan större anpassningar, dels för att Perl är speciellt lämpat för de ganska enkla textfils- och textsträngshanteringar som det oftast är frågan om. Andra populära språk för att skriva CGI-program är shellscript, C, C++ och VisualBasic. Exemplen nedan är skrivna i Perl.

Oftast ska CGI-program ligga i en speciell cgi-bin-mapp för att servern ska veta att den ska köra (exekvera) programmen, och inte skicka filerna som vanligt. En del servrar går också att ställa in så att det går att köra CGI-program från vilken mapp som helst. I det fallet brukar filerna ha namn som slutar på **.cgi**. På Unix-servrar måste CGI-programmen dessutom göras executable med chmod för att fungera.

Slarvigt skrivna CGI-program kan öppna allvarliga säkerhetshål i webbservern. Webbhotell och internetleverantörer brukar därför i allmänhet ha särskilda restriktioner för CGI. En del tillåter inte alls CGI, andra vill kontrollera alla program innan de installeras. Några få webbhotell låter kunderna själva installera CGI-program. Kontrollera med ditt webbhotell vilka regler som gäller.

En sida som skapats av ett CGI-program har, precis som alla andra sidor, en webbadress, till exempel http://www.firma.se/cgi-bin/allan. När webbläsaren anropar servern och begär att få denna sida kommer alltså programmet ”allan” att köras igång på serverdatorn, och utdata från det programmet (som t.ex. kan vara HTML-kod) skickas till webbläsaren.

Det är vanligt att man kommer till en CGI-genererad sida efter att ha fyllt i ett formulär. Sidans adress anges i formulärets ACTION-attribut. Oftast beror sidans utseende då på vad användaren fyllt i i formuläret. Man kan också komma till en CGI-genererad sida via en vanlig länk.

Att skriva CGI-program kräver kunskaper i programmering, och detta ligger utanför den här handledningen. Men jag ska ändå ge några upplysningar. Ett CGI-program måste alltid ge ifrån sig en eller flera s.k. HTTP-headers. Dessa kan bland annat ange vilken typ av data (mediatyp) som CGI-programmet ger till webbläsaren, eller att webbläsaren ska dirigeras om till en annan sida. Efter sista header-raden kommer alltid en blankrad, och i förekommande fall själva sidans innehåll (till exempel HTML-kod).

Vanliga header-rader när man skriver egna CGI-program är till exempel:

Content-type: text/html Anger mediatypen (kallas även MIME-typ) och det är den som styr hur webbläsaren ska tolka CGI-programmets utdata. Är mediatypen text/html vet webbläsaren att den ska tolka HTML-koder, är mediatypen text/plain så kommer texten visas precis som den är, om mediatypen är image/gif så vet webbläsaren att filen är en GIF-bild etc.

Location: http://www.server.se/mapp/fil.html Gör så att webbläsaren dirigeras om och laddar in det angivna dokumentet. Adressen anges som en absolut URL, d.v.s. börjar med http. Adressen i webbläsarens adressfält kommer att ändras till den nya adressen.

Location: /mapp/fil.html Gör så att servern, istället för den begärda filen, skickar det angivna dokumentet. Filnamnet måste anges med full sökväg, d.v.s. börja med / Adressen i webbläsarens adressfält kommer *inte* att ändras.

Status: 204 No Response Anger att webbläsaren inte ska vänta på data från servern. Detta används ifall ett CGI-program inte ska producera någon webbsida, utan göra något annat. Webbläsaren blir kvar på den sida den var och gör ingenting.

Refresh: n Anges på en extra rad efter Content-type och tvingar webbläsaren att ladda om dokumentet efter n sekunder (se avsnittet om Client Pull).

Refresh: n; URL=http://foo.bar/blatz.html Anges på en extra rad efter Content-type och tvingar webbläsaren ladda in dokumentet http://foo.bar/blatz.html efter n sekunder (se avsnittet om Client Pull).

Window-target: window_name Anges på en extra rad efter Content-type och tvingar webbläsaren att öppna dokumentet i ett fönster med angivet namn (se avsnittet om FRAME och TARGET). Om det inte finns något fönster med det namnet så kommer ett nytt att öppnas.

Set-Cookie: customer="768a5f"; path="/shop"; domain="firma.se" En s.k. magic cookie som skickas från servern (se avsnittet om Magic Cookies).

Alla dessa HTTP-headers ska ges på egna rader, och efter den sista av dem ska det alltid komma en blankrad innan innehållet som ska visas på sidan kommer. En blankrad ska alltid skickas, även om det inte kommer något mer data (som efter Location eller Status: 204 No Response).

Några enkla exempel skrivna i Perl:

Detta program genererar en sida med texten Hello World

```
#!/usr/local/bin/perl
print "Content-type: text/plain\n\n";
print "Hello World";
```

Notera att mediatypen är text/plain, d.v.s. ren, oformaterad text. Efter headerraden kommer två ny rad-tecken, dvs det blir en blank rad mellan HTTP-headern och texten.

Detta program genererar också en sida med texten Hello World

```
#!/usr/local/bin/perl
print "Content-type: text/html\n\n";
print "<TITLE>Hello World</TITLE>Hello World";
```

Notera att mediatypen är text/html, d.v.s. HTML-kod, och att sidan har en titel.

Detta program dirigerar om webbläsaren till olika sidor beroende på om användaren kommer från en svensk domän eller ej. Indata tas från environment-variabeln REMOTE_HOST som innehåller namnet på anropande dator, och utdata är en Location-header.

```
#!/usr/local/bin/perl
$land = substr ($ENV{'REMOTE_HOST'},
        rindex ($ENV{'REMOTE_HOST'}, "."));
if ($land eq ".se") { print "Location: /svensk-sida.html\n\n"; }
else { print "Location: /engelsk-sida.html\n\n"; }
```

Environment-variabler

Webbservern sätter ett antal environment-variabler som CGI-programmet kan läsa in och använda. Vilka variabler som kan användas skiljer sig något mellan olika servrar, men dessa är vanligt förekommande:

Dessa environment-variabler har hela tiden samma värde på en viss server:
SERVER_SOFTWARE Webbserverns programvarunamn och version.
SERVER_NAME Webbserverdatorns DNS-namn.
SERVER_ADMIN E-postadress till serverns administratör.
GATEWAY_INTERFACE Version av CGI-specifikationen som servern stöder.
DOCUMENT_ROOT Fysisk sökväg till den mapp på servern där HTML-filerna ligger.

Dessa environment-variabler har olika värden beroende på klientens begäran om sidan:
SERVER_PROTOCOL Version av HTTP-protokollet som begäran kom in med.
SERVER_PORT IP-porten som begäran kom in till (vanligen port 80).
REQUEST_METHOD Den metod som begäran görs med (GET, POST eller HEAD).
REQUEST_URI Den virtuella sökväg som begäran görs till.
PATH_INFO Extra path-information från klienten (kan användas för att ge programmet extra information via URL:en).
PATH_TRANSLATED Samma som PATH_INFO, plus sökvägen till server root.

SCRIPT_NAME Virtuell sökväg till CGI-programmet.
SCRIPT_FILENAME Fysisk sökväg till CGI-programmet.
QUERY_STRING Det som står efter ? i URL:en programmet anropades med.
REMOTE_HOST Anropande dators namn (om servern gör namnuppslagning).
REMOTE_ADDR Anropande dators IP-nummer.
REMOTE_PORT Anropande dators portnummer.
AUTH_TYPE Autentisieringsmetod för lösenordsskyddade CGI-program.
REMOTE_USER Användarnamn som angetts vid en eventuell lösenordsruta.
REMOTE_IDENT Användarnamn enligt RFC 931 (ovanligt).
CONTENT_TYPE Mediatyp för det datablock med fomulärdata som sänds från webbläsaren.
CONTENT_LENGTH Längd på datablock med fomulärdata som sänds från webbläsaren.

Alla HTTP-header-rader som klienten skickar till servern vid begäran om sidan läggs in som environment-variabler med namn som börjar på HTTP_ följt av headerns namn. Bindestreck i header-namnet ändras till understreck. Exempel på variabler som kan finnas (beror på klienten):
HTTP_ACCEPT En lista på de mediatyper som webbläsaren kan hantera.
HTTP_ACCEPT_CHARSET En lista på de teckenuppsättningar som webbläsaren kan hantera.
HTTP_ACCEPT_LANGUAGE De språk som användaren föredrar.
HTTP_COOKIE Magic cookie som skickas från webbläsaren (se detta avsnitt).
HTTP_HOST Det datornamn som begäran riktades till.
HTTP_REFERER Adressen till den webbsida som användaren länkar sig ifrån.
HTTP_USER_AGENT Webbläsarens programvaruversion.

SSI – Server-side includes

Med hjälp av server-side includes kan ett dokument infogas i ett annat. När användaren klickar fram en ny sida så skickar servern normalt en HTML-fil till användarens webbläsare. Om man använder server-side includes så söker servern igenom HTML-filen efter s.k. include-direktiv innan den skickar iväg den, och byter ut direktiven mot extra HTML-kod som alltså infogas ”i farten” i den HTML-fil som användaren begärde.

Den HTML-kod som infogas kan antingen vara ett komplett HTML-dokument på serverns hårddisk, eller vara utdata från ett CGI-program. Det kan också vara till exempel anropande dators IP-nummer, aktuell tid, datum då ett dokumentet senast uppdaterades etc.

Att söka igenom alla dokument efter include-direktiv tar extra tid och datorkraft av servern, så därför brukar man ställa in servern så att den bara söker igenom de dokument där webbdesignern vet att det finns include-direktiv. Den vanligaste inställningen är att servern söker igenom filer som istället för .html slutar på **.shtml**. Andra ändelser som förekommer på vissa servrar är **.sml**, **.sht** och **.ssi**.

Ett include-direktiv har utseendet **<!--#command item="value"-->** Det ser alltså ut som en vanlig HTML-kommentar, det vill säga en textsträng omgiven av <!-- och -->. Detta gör att include-direktivet inte påverkar sidans utseende ifall det av någon anledning inte skulle behandlats av servern innan filen skickades, till exempel när man tittar på sidan lokalt från sin hårddisk, eller om webbservern inte stöder SSI.

Server-side includes är en funktion hos webbservern, och exakt vilka kommandon i include-direktiven som stöds varierar från server till server.

Webbhotell och internetleverantörer brukar i allmänhet ha särskilda restriktioner för SSI. De flesta tillåter inte alls SSI, andra tillåter endast include, och inte exec. Kontrollera med ditt webbhotell vilka regler som gäller.

Några av de vanligaste include-direktiven är:

<!--#exec cgi="/cgi-bin/filnamn"--> Exekverar ett CGI-program och ersätter include-direktivet med utdata från programmet.

<!--#exec cmd="unix-kommando"--> Öppnar ett shell (/bin/sh) och exekverar angivet Unix-kommando. Eventuell utdata läggs in i sidan.

<!--#flastmod file="dokument.html"--> Ersätter include-direktivet med datum för senaste redigeringen av det angivna dokumentet. Datumformatet kan styras med include-direktivet <!--#config timefmt="%Y-%m-%d klockan %R"-->.

<!--#fsize file="dokument.html"--> Ersätter include-direktivet med storleken på den angivna filen. Formatet kan styras med include-direktiven <!--#config sizefmt="abbrev"--> och <!--#config sizefmt="bytes"-->.

<!--#include file="dokument.html"--> Ersätter include-direktivet med innehållet i en fil på servern. Filnamnet anges relativt nuvarande mapp. Det går inte att ange filer uppåt i filstrukturen.

<!--#include virtual="/mapp/dokument.html"--> Ersätter include-direktivet med innehållet i en fil på servern. Filnamnet anges som absolut path (måste börja med /).

<!--#echo var="env-var"--> Ersätter include-direktivet med värdet på en s.k. environment-variabel. Man kan använda samma environment-variabler som för CGI (se detta avsnitt) samt ytterligare några, till exempel **DOCUMENT_NAME**, **DOCUMENT_URI**, **QUERY_STRING_UNESCAPED**, **DATE_LOCAL**, **DATE_GMT** och **LAST_MODIFIED**.

Magic Cookies

Magic Cookies är en metod för en server eller ett CGI-program att spara information som rör en viss användare hos den användaren. När användaren går in på en viss sida så ger servern användarens webbläsare en ”kaka”, en Magic Cookie, som webbläsaren sparar i sin ”kakburk”, en fil på användarens hårddisk. Nästa gång som användaren besöker samma sida lämnar webbläsaren ifrån sig kakan till servern som kan basera sitt agerande på innehållet i den.

Kakan är en rad text som innehåller ett NAME/VALUE-par, samt ofta information om till vilken server och vilken sida på servern som kakan ska ges när användaren går dit. Dessutom kan kakan innehålla ett datum och/eller tid då den inte gäller längre.

Kakor kan användas för alla tillämpningar när man vill hålla reda på vem användaren är från gång till gång hon besöker ens sidor. Ett klassiskt exempel är ”shoppingvagnen” där användaren går från sida till sida och samlar ihop de varor hon vill ha, och till slut beställer dem på en beställningssida. Information om vilka varor hon valt ut kan då lagras i en kaka som skickas av webbläsaren till servern när användaren går in på beställningssidan.

Andra exempel är att användaren första gången hon går in på en flerspråkig webbtjänst väljer vilket språk hon föredrar, och att den informationen sparas i en kaka som ges till servern varje påföljande gång hon går dit, så att servern ”automagiskt” kan välja rätt språk. Man kan också spara uppgifter som kundnummer, användarnamn och lösenord i en kaka, så att besökaren slipper skriva in dem på nytt. Man ska dock vara försiktig med att lita alltför mycket på kakor som besökarnas webbläsare ger tillbaka till servern, eftersom de lagras i en textfil som förslagna personer lätt kan redigera.

Man ska också vara medveten om att många användare av integritetsskäl ogillar den möjlighet att registrera sina besökare som kaktekniken ger, och därför har slagit av funktionen i sina webbläsare.

Servern eller ett CGI-program ger kakan till webbläsaren genom en extra HTTP-header-rad:

```
Set-Cookie: NAME=VALUE; expires=DATE; domain=DOMAIN; path=PATH; secure
```

Webbläsaren sparar kakan i sin ”kakburk”, och nästa gång som användaren klickar fram den sida med ett namn som matchar PATH på en server vars namn matchar DOMAIN och tiden DATE inte har gått ut kommer webbläsaren ge kakan till servern genom en extra rad i sin begäran att få se sidan. Raden ser ut som:

```
Cookie: NAME=VALUE
```

Har webbläsaren flera kakor som matchar kriterierna ovan ger den alla på samma rad:

```
Cookie: NAME1=VALUE1; NAME2=VALUE2
```

NAME och **VALUE** kan vara vilken text som helst som inte innehåller semikolon, komma eller mellanslag. NAME och VALUE är det enda som måste vara med i en Set-Cookie.

DATE är det datum och tid som kakan ska sluta gälla. Strängen formateras som i

```
Wednesday, 09-Nov-99 23:12:40 GMT
```

Om DATE inte anges gäller kakan tills användaren stänger av sin webbläsare.

DOMAIN är namnet på den server som kakan ska ges till vid nästa besök. Namnet matchas bakifrån så att **.firma.se** både matchar servrarna order.firma.se och shopping.korg.firma.se. Bara servrar som själva matchar DOMAIN kan ge kakan, och DOMAIN måste innehålla två punkter (det går till exempel inte att ha DOMAIN=".se").

Om DOMAIN inte anges sätts den till samma som den server som gav kakan.

PATH är adressen till den sida som kakan ska ges till. PATH matchas framifrån, så att ”/foo” matchar både ”/foobar” och ”/foo/bar.html”. PATH="/" matchar alla sidor på servern.

Om PATH inte anges sätts den till samma som den sida som gav kakan.

secure anger att kakan bara får skickas över säkra förbindelser, d.v.s. kakan kommer bara att skickas om förbindelsen mellan server och klient är krypterad med SSL (s.k. https-servrar).

Ett CGI-program kan ta in den kaka som webbläsaren skickat genom att läsa av environment-variabeln HTTP_COOKIE.

Läs specifikationen till Magic Cookies på http://home.netscape.com/newsref/std/cookie_spec.html

Client-side scripts

Ett client-side script är ett dataprogram som körs på användarens dator. Programmet kan antingen ligga inbäddat i HTML-koden eller anropas via en script-deklaration (se nedan). Programmet körs antingen igång när dokumentet laddas in, eller vid något senare tillfälle, som till exempel när musen rörs över ett visst objekt på sidan, eller när användaren klickar på en länk.

Client-side scripts kan bland annat användas för att dynamiskt ändra innehållet på webbsidan (lägga till text, byta ut bilder etc), processa formulärinnehåll (fylla i formulärfält beroende på värdet i andra fält, kontrollera att ifyllda värden ligger inom ett tillåtet intervall etc), göra dialogrutor som kommer upp när användaren klickar på en knapp, skriva saker i webbläsarens statusrad (det utrymme som normalt används för att visa pågående inladdning samt vart länkar leder), låta webbsidan förändras när användaren för musen över vissa ytor på sidan etc.

HTML-specifikationen medger att man lägger in eller anropar program skrivna i vilket programmeringsspråk som helst, men för att programmet ska kunna köras krävs att användarens webbläsare har stöd för det språk som används. Exempel på programmeringsspråk som används i detta sammanhang är Javascript, Vbscript och Tcl.

Javascript

Javascript (som tidigare hette Livescript) är ett programmeringsspråk för client-side scripts som stöds av flera olika webbläsare, bland annat Netscape Communicator och nyare versioner av Microsoft Internet Explorer. Exemplen här nedan är skrivna i Javascript.

En varning kan vara på sin plats här: de webbläsare som stöder Javascript har var och en sin egen uppfattning om hur språket skall tolkas, och alla olika versioner av dem (Windows, Macintosh, Unix etc) har sin egen uppsättning programfel. Så bered dig på mycket testande; ett Javascript som fungerar utmärkt i en webbläsarversion kan göra att en annan webbläsare och/eller användarens dator låser sig.

Javascript är *inte* detsamma som Java.

Scripts som körs direkt

De client-side scripts som ska köras igång direkt när laddas in läggs mellan styrkoderna **<SCRIPT>** och **</SCRIPT>**. Programmet kan ligga antingen inom ”head”- eller ”body”-delen av HTML-filen.

Man kan lägga in scriptet i HTML-koden, då skriver man

```
<SCRIPT TYPE="text/javascript">
...javascriptets kod och kommandon...
</SCRIPT>
```

Man kan också anropa ett script i en annan fil med

```
<SCRIPT TYPE="text/javascript" SRC="http://www.firma.se/script/allan">
</SCRIPT>
```

Notera att man måste ange vilket scriptspråk som används med **TYPE**. Som värde till TYPE anges en giltig mediatyp, i detta fall "text/javascript". Tidigare var det vanligt att man angav scriptspråket med attributet **LANGUAGE** istället för TYPE, men detta rekommenderas inte, eftersom de värden som LANGUAGE kan ha inte är standardiserade.

Scripts som körs vid speciella händelser

De client-side scripts som ska köras igång när en speciell händelse inträffar anropas med speciella händelse-attribut (intrinsic events) som läggs till den styrkod som har med händelsen att göra.

De händelse-attribut som finns är **onload**, **onunload**, **onclick**, **onbdlclick**, **onmousedown**, **onmouseover**, **onmousemove**, **onmouseout**, **onfocus**, **onblur**, **onkeypress**, **onkeydown**, **onkeyup**, **onsubmit**, **onreset**, **onselect** och **onchange**.

Om man till exempel vill att ett script ska köras när muspekaren rörs över en bild lägger man in scriptet med attributet onmouseover till <IMG>-koden. Om man vill att ett annat script ska köras när muspekaren rörs bort ifrån bilden lägger man in detta med attributet onmouseout.

Ett exempel på en vanlig tillämpning är att styra vad som kommer visas i statusraden längst ner i webbläsarfönstret när användaren pekar på en länk (men inte klickar). Man lägger då in onmouseover som ett attribut till styrkoden <A HREF>:

```
<A HREF="kaka.html" onmouseover="window.status='Allan tar kakan';
return true">Länktexten</A>
```

När man för muspekaren över länken kommer texten ”Allan tar kakan” visas i statusraden istället för den vanliga informationen om vart länken leder.

Webbläsare som inte stöder scripts

För att erbjuda ett alternativt innehåll för webbläsare som inte stöder det scriptspråk som används, och för webbläsare som inte alls stöder client-side scripts, kan man använda styrkoden **<NOSCRIPT>**. Koden avslutas med **</NOSCRIPT>**.

Webbläsare som inte stöder client-side scripts alls kommer förmodligen att visa script-koden mellan <SCRIPT> och </SCRIPT> som text på webbsidan. För att undvika detta så kan man använda kommentarskoder för att dölja koden för dessa. Kommentaren börjar med <!-- som vanligt. Denna kod bör stå ensam på raden. Kommentaren ska sluta med -->, men för att inte den koden ska tolkas som att den ingår i scriptet så måste den föregås av script-språkets kommentars-kod. För Javascript är den //, dvs kommentaren slutar med en rad med koden // -->. Andra scriptspråk använder andra kommentarsmarkeringar.

Läs mer om client-side scripts på
http://www.w3.org/TR/WD-html40/interact/scripts.html

Läs mer om Javascript på
http://monkeytoys.com/javascript/
http://home.netscape.com/eng/javascript/index.html
http://ls.ctc.edu/Exp/Technique/Java/

Ljud och film

Förutom text, stillbilder och animerade GIF-bilder går det även att ha ljud, film och andra typer av animeringar på sin webbsida. Precis som för bilder finns det en mängd olika filformat för ljud och film som alla har sina olika egenskaper och kompromisser mellan kvalitet och filstorlek. Tyvärr finns det inget ”universellt” filformat som man kan vara någorlunda säker på att de flesta användares datorer och program kan hantera, som GIF för stillbilder. Dessutom så kan inte alla datorer alls spela upp ljud.

De vanligaste ljudfilsformaten på webben idag är Sun Audio (filerna slutar på **.au** eller **.snd**), Windows WAVE (filerna slutar på **.wav**) och Apples Audio Interchange File Format (filerna slutar på **.aif**, **.aiff** eller **.aifc**). Andra format som förekommer är Soundblaster VOC, Amiga IFF, Amiga MOD, MIDI och MPEG.

De vanligaste filmfilsformaten på webben idag är Apples QuickTime (filerna slutar på **.qt** eller **.mov**), MPEG från Moving Pictures Expert Group (filerna slutar på **.mpeg**, **.mpg** eller **.mpe**) och Windows Audio-Video Interleaved (filerna slutar på **.avi**). Ett annat format som förekommer är Autodesk FLI/FLC animation som lämpar sig speciellt för dataanimerade filmer (filerna slutar på **.flc** eller **.fli**).

Länk till ljud eller film

Det traditionella sättet att spela ljud eller visa film på en webbsida är att göra en vanlig länk med <A HREF="ljudfil.au">. Länken leder till en ljud- eller filmfil i lämpligt format som hämtas till webbläsaren när användaren klickar på länken. Webbläsaren lämnar då över filen till en s.k. helper application, ett annat program i användarens dator som kan spela ljudet eller visa filmen. Detta kräver alltså att användaren dels har ett sådant program som kan hantera denna filtyp, dels att hon ställt in sin webbläsare riktigt.

Ifall man har länkar till ljud- eller filmfiler på sin sida kan det vara en bra idé att också tala om vilket hjälpprogram som användaren behöver (både för Windows, Macintosh och Unix) samt erbjuda en länk till ett filarkiv där man kan hämta detta program.

Tänk också på att se till att servern skickar dina ljud- och filmfiler med rätt mediatyp angiven (ställs in i serverns inställningar), annars vet inte användarens webbläsare hur den ska hantera dem.

Ljud eller film på själva sidan

Med de nyare versionerna av de populära webbläsarna kan man placera ljud och film som integrerade delar på själva webbsidan, precis som med vanliga stillbilder. För varje ljud på en sida lägger webbläsaren då ut en start-, en paus- och en stoppknapp samt en volymkontroll, och filmer visas i en ruta med motsvarande kontroller under eller vid sidan om.

Vissa ljud- och filmformat kan webbläsaren själv hantera på detta sätt, för andra krävs att användaren har en s.k. plug-in installerad. Om rätt plug-in inte finns installerad kan en del webbläsare själva ge förslag på vilken plug-in användaren måste skaffa, men även här kan det vara en bra idé att erbjuda en länk till ett filarkiv där man kan hämta en lämplig plug-in. Även här måste du se till att servern skickar dina ljud- och filmfiler med rätt mediatyp angiven.

För att lägga in ljudet eller filmen på webbsidan använder man styrkoden **<OBJECT>**. Till <OBJECT> lägger man diverse attribut som styr hur ljudet eller filmen presenteras på sidan, samt information som den plug-in eller yttre program som ska spela upp filen behöver för att fungera

riktigt. Exempel på dessa attribut är **codebase**, **classid**, **codetype**, **data**, **type**, **declare**, **standby** och **align**. Om uppspelningsprogrammet behöver mer information kan den ges med attributen **name**, **value**, **valuetype** och **type** till styrkoden **<PARAM>**. Exakt hur man gör beror på det ljud- eller filmfilsformat samt det uppspelningsprogram som används. Mer information brukar finnas hos de företag som tillverkar konverterings- och uppspelningsprogram för de olika formaten.

Om webbläsaren inte kan hantera den datatyp som anges i en <OBJECT>-kod kommer den att försöka visa det som finns mellan <OBJECT> och </OBJECT>. Mellan start- och slutkod kan man då lägga underordnade <OBJECT>-koder för att erbjuda samma innehåll i olika format:

```
<OBJECT CLASSID="TheEarth.py">
<OBJECT DATA="TheEarth.mpeg" TYPE="application/mpeg">
<OBJECT SRC="TheEarth.gif">
Jorden sedd från rymden.
</OBJECT>
</OBJECT>
</OBJECT>
```

I detta exempel kommer webbläsaren i första hand att visa en animation skriven i programmeringsspråket Python. Om webbläsaren inte kan det, visas en MPEG-film istället. Kan webbläsaren inte det heller, visas en GIF-bild, och går inte ens det så visas texten.

Netscapes metod

Netscape version 3 och 4 stöder <OBJECT> endast i begränsad omfattning. Netscapes metod är istället att man lägger in ljud- och filmfiler med **<EMBED>**. Man kan lägga in ett alternativt innehåll för de webbläsare som inte stöder <EMBED> mellan styrkoderna **<NOEMBED>** och **</NOEMBED>**.

Till <EMBED> lägger man olika attribut som styr hur ljudet eller filmen presenteras på webbsidan. Vissa, som WIDTH och HEIGHT används av Netscape självt för att styra hur ljudets eller filmens ruta på webbsidan ska se ut, medan andra attribut vidarebefordras till den plug-in som hanterar filen. Vilka attribut som kan användas beror på den plug-in som används för just detta filformat. Några av de vanligaste är:

SRC="URL" Adressen till ljud- eller filmfilen. Anges som relativ eller absolut adress, på samma sätt som alla andra URL:er.

AUTOSTART="TRUE" gör att ljudet/filmen börjar spelas upp automatiskt så fort det har laddats in. Anges inte AUTOSTART, eller sätts **AUTOSTART="FALSE"** laddas ljudet/filmen in, men spelas inte upp förrän användaren klickar igång det.

CONTROLS="CONSOLE | SMALLCONSOLE | PLAYBUTTON | PAUSEBUTTON | STOPBUTTON | VOLUMELEVER" anger vilken typ av kontrollpanel som ska synas. Om CONTROLS inte anges visas en CONSOLE.

HIDDEN="TRUE" gör att ingen kontrollpanel visas, och att ljudet spelas som ett bakgrundsljud. Undvik att kombinera detta med oändlig loopning av ljudet.

LOOP="TRUE" gör att uppspelningen börjar om från början när filmen/ljudet är slut. Uppspelningen fortsätter tills användaren klickar på stopp eller lämnar sidan. Anges inte LOOP, eller sätts **LOOP="FALSE"** spelas ljudet/filmen upp en gång. För ljudfiler kan man även ange en siffra, till exempel **LOOP="5"**, vilket gör att ljudet spelas upp det antalet gånger i följd.

WIDTH="n" och **HEIGHT="n"** anger storleken på ljuduppspelningens kontrollpanel (start- och stoppknapparna) eller på filmrutan. n anges i pixlar. Standardstorleken på en CONSOLE är 144x60. Standardstorleken på en SMALLCONSOLE är 144x15. Standardstorleken på en VOLUMELEVER är 74x20. Standardstorleken på en ljudknapp är 37x22. Vanliga storlekar på filmer är 90x120, 120x160, 180x240 och 240x320.

ALIGN bestämmer hur text flyter runt ljudkontrollerna eller filmrutan. ALIGN fungerar på samma sätt här som inom styrkoden <IMG>.

Med **STARTTIME="min:s"** och **ENDTIME="min:s"** kan man ange var i en ljudfil som uppspelningen ska börja respektive sluta. Anges inte STARTTIME eller ENDTIME så spelas ljudet upp från början till slutet.

VOLUME="50" anger ett ljuds uppspelningsvolym i procent. Värdet ska vara mellan 0 och 100. Om VOLUME inte anges spelas ljudet upp med tidigare systeminställning.

Om man har flera <EMBED>-koder med lösa kontrollobjekt (start-, och stoppknapp, volymreglage etc) som alla ska kontrollera samma ljud så sätter man attributet **MASTERSOUND** i den <EMBED>-kod som innehåller ljudet, och i alla de <EMBED>-koder som hör ihop sätter man **NAME**-attribut med samma namn. Om ett volymreglage ska kontrollera flera NAME-grupper (eller systemvolymen) används **NAME="MASTERVOLUME"**.

Några exempel:

```
<EMBED SRC="filmer/lokal.mov" WIDTH="200" HEIGHT="150">
<EMBED SRC="melodi.wav" WIDTH="144" HEIGHT="60" AUTOSTART="TRUE"
LOOP="TRUE">
<EMBED SRC="vd-tal.aif" AUTOSTART="FALSE" VOLUME="100" WIDTH="144"
HEIGHT="60" CONTROLS="CONSOLE">
```

Ljud i andra webbläsare än Netscape

Microsoft Internet Explorer kan spela bakgrundsljud med den egna styrkoden <BGSOUND SRC="allan.wav">. Nyare versioner av Mosaic stöder koden <SOUND SRC="allan.wav">. Dessa koder anges varsomhelst i <BODY>-delen av dokumentet.

En annan variant som fungerar för alla webbläsare som stöder omdirigering med <META HTTP-EQUIV> är att i <HEAD>-delen av dokumentet lägga <META HTTP-EQUIV="Refresh" CONTENT="0; URL=ljudfil.wav">. Detta gör att webbläsaren direkt efter att sidan laddats in hämtar ljudfilen.

Läs mer om Netscapes ljud- och film-plug-ins på
http://home.netscape.com/comprod/products/navigator/version_3.0/multimedia/audio/how.html
http://home.netscape.com/comprod/products/navigator/version_3.0/multimedia/video/how.html
Läs mer om olika ljudformat på
http://www-cs-students.stanford.edu/~franke/SoundApp/formats.html
Läs mer om MPEG på http://www.cis.ohio-state.edu/hypertext/faq/usenet/mpeg-faq/top.html och http://www.powerweb.de/mpeg/mpegfaq/
Läs mer om QuickTime på http://quicktime.apple.com/
Läs mer om Autodesk FLI/FLC på http://vineyard.er.usgs.gov/movies/movies/flc.html

RealAudio, RealVideo, Vivo och Streamworks

Med de metoder som beskrivits ovan så börjar uppspelningen av ljudet eller filmen först när hela filen förts över till användarens dator. Detta kan ta lång tid eftersom ljud- och filmfiler ofta är stora. Därför finns det även metoder för att spela upp ljudet eller filmen i realtid allteftersom den kommer över nätet, s.k. streaming audio respektive streaming video. De mest använda är RealAudio och Streamworks för ljud, och RealVideo och Vivo för film.

RealAudio och RealVideo fungerar så att ljud- eller filmfilen levereras av ett separat serverprogram. Från webbsidan har man en <A HREF>-länk till en s.k. metafil som innehåller adressen till RealAudio-servern. Webbläsaren lämnar över adressen till ett hjälpprogram som själv tar kontakt med RealAudio-servern. Den fortsatta kontakten dem emellan sker alltså inte via webbläsaren, och inte ens via HTTP, utan via en egen kanal över nätet, som är mer avpassad för streamad media.

Streamworks, som är en konkurrerande teknik, fungerar på liknande sätt.

Vivo-filmer kräver ingen speciell server, utan kan levereras av den vanliga webbservern. En fördel med detta är att man slipper ha en speciell server installerad.

Läs mer om RealAudio och RealVideo på http://www.real.com/
Läs mer om Streamworks på http://www.streamworks.com/
Läs mer om Vivo på http://www.vivo.com/

Multimedia

I datasammanhang brukar multimedia betyda att man har två eller flera ”medier” (text, stillbild, tal, musik, film och animering etc) närvarande samtidigt. Ofta kan också användaren navigera i materialet, välja vilka bilder hon vill se och vilka ljud hon vill höra. Man kan säga att world wide web är ett multimedia-system eftersom man kan ha olika ”medier” på en webbsida eller grupp av flera webbsidor.

Det finns också andra system för multimedia än world wide web, som är utvecklade för andra miljöer än Internet, till exempel presentationsmaterial på cd-rom. Man använder speciella programvaror för att skapa och visa dessa. Med hjälp av tilläggsmoduler till webbläsarna kan man numera kombinera systemen och lägga in sådana presentationer på sina webbsidor.

Shockwave

Shockwave är en tilläggsmodul till Netscape, en s.k. plugin, som gör det möjligt att spela upp multimediapresentationer, animationer och vektorgrafikbilder skapade i programmen Macromedia Director, Macromedia Flash och Macromedia Authorware direkt i webbläsarens fönster.

Macromedia Director är ett program som används för att göra multimediapresentationer på cd-rom. Antingen länkar man till sin Director-presentation med <A HREF="presentation.dcr"> (då visas presentationen ensam i Netscape-fönstret) eller så lägger man in presentationen som en integrerad del av webbsidan med <EMBED SRC="presentation.dcr" WIDTH="496" HEIGHT="350">.

För att kunna se en Director-presentation på en webbsida måste användaren ha en Shockwave-plugin installerad. Ofta krävs också att användaren tilldelar Netscape några megabyte mer arbetsminne.

Director-filer som visas med Shockwave har namn som slutar på .dcr eller .dir

Läs mer om Shockwave: http://www.macromedia.com/shockwave/

Adobe Acrobat

Adobe Acrobat är ett elektroniskt publiceringssystem utvecklat av Adobe. Det bygger på sidbeskrivningsspråket PostScript, också det utvecklat av Adobe. Normalt när man ska skriva ut ett dokument på en laserskrivare så skapar datorn en PostScript-fil som sedan sänds till skrivaren. Med Acrobat konverterar man istället denna fil till en .pdf-fil (Portable Document Format) och kan sedan titta på filen med programmet Acrobat Player.

Det finns även versioner av Acrobat Player som fungerar som s.k. plug-ins till Netscape, och på så vis kan .pdf-filer visas direkt i Netscapes fönster. Acrobat-systemet gör det lätt att publicera en tryckt skrift på webben genom att helt enkelt spara den som .pdf-filer. Fördelen med detta är att sidornas exakta utseende bevaras, med typsnitt, färger, bilder och format intakta.

Med de mer avancerade programmen i Adobe Acrobat-familjen kan man också skapa hypertextlänkar inne i .pdf-dokument så att användaren kan navigera runt inne i den virtuellt tryckta skriften.

Läs mer om Adobe Acrobat på http://www.adobe.com/acrobat/

VRML

VRML står för Virtual Reality Modeling Language. Med VRML kan man skapa interaktiva, tredimensionella bilder som antingen kan läggas in på webbsidor eller visas fristående. Ett VRML-dokument brukar kallas värld och för att kunna se webbsidor med VRML-världar måste användaren ha en speciell plug-in till Netscape.

Typiska användningsområden är att användaren kan rotera ett objekt så att hon kan se det från alla sidor eller att användaren kan ”gå omkring” i ett rum eller ett landskap.

VRML-filer har namn som slutar på .wrl

Läs mer om VRML på http://vag.vrml.org/

Andra datatyper

Det finns en mängd olika tekniker, system och filformat som kan läggas in som objekt på webbsidor med <EMBED>, <OBJECT> eller motsvarande styrkoder. Man bör vara försiktig med att använda dem, eftersom användaren måste skaffa hjälpprogram eller plug-ins för varje nytt filformat som hon stöter på. Alla plug-ins finns inte tillgängliga för alla datorsystem, och kan ofta vara illa skrivna, vilket kan medföra att webbläsaren och/eller användarens dator låser sig.

En sammanställning över många plug-ins finns på http://browserwatch.internet.com/plug-in.html Nyare versioner av Netscape föreslår själv vilka plug-ins som kan behövas till en viss, ny datatyp.

Java

Java är ett programmeringsspråk som liknar C och C++. Java introducerades av Sun och målet är att Java-program ska vara helt plattformsoberoende och gå att köra på alla datorer. Med Java gör man aningen fullständiga program, applications, eller små programkomponenter, applets, som kan interagera med andra program och applets, både i användarens dator och i andra datorer på nätet.

En Java-applet hämtas från webbservern och körs sedan på användarens maskin. Användaren måste ha en webbläsare som stöder Java. Java-applets kan vara fristående eller ligga infogade på en webbsida med styrkoden <OBJECT> eller **<APPLET>**.

Exempel med <APPLET> (fungerar i de flesta webbläsare, inkl. Netscape):

```
<APPLET CODE="Bubbles.class" WIDTH="500" HEIGHT="500">
Java-applet som ritar animerade bubblor.
</APPLET>
```

Exempel med <OBJECT> (rekommenderas enligt HTML-specifikationen):

```
<OBJECT CODETYPE="application/octet-stream"
CLASSID="java:Bubbles.class"
WIDTH="500" HEIGHT="500">
Java-applet som ritar animerade bubblor.
</OBJECT>
```

Java och Javascript är *inte* samma sak.

Läs mer om Java
Sun Microsystems: http://java.sun.com/
Gamelan: http://www.gamelan.com/
Webbtidningen JavaWorld: http://www.javaworld.com/

Språk och textriktning

Med attributet **LANG** kan man specifiera vilket språk som texten på sidan är skriven i. Detta kan bland annat utnyttjas av sökmaskiner, talsyntesewebbläsare och rättstavningsprogram. Det kan också användas av grafiska webbläsare för att välja den typ av citattecken och styckesindelningar som används i det aktuella språket.

Attributet kan anges till styrkoden <HTML> och gäller då hela dokumentet, eller till någon annan styrkod, t.ex. <P>, och gäller då enbart för det stycket (användbart om man har kortare stycken med ett annat språk i sin text).

Språket anges med en kod enligt RFC 1766 (se ftp://ds.internic.net/rfc/rfc1766.txt) och består oftast av två bokstäver enligt ISO 639 (http://www.iso.ch/cate/d4766.html) för att ange språket, och eventuellt ett bindestreck samt en landskod enligt ISO 3166 eller annan kod för att ange dialekt eller språkvariant.

Exempel:

LANG="en"	Engelska
LANG="en-us"	Amerikansk engelska
LANG="en-cockney"	Cockney-engelska
LANG="sv"	Svenska
LANG="sv-fi"	Finlandssvenska

I vissa språk skriver man från vänster till höger, och i andra skriver man från höger till vänster. För att en webbläsare ska kunna rita upp texten på skärmen på rätt sätt kan man ange attributet **DIR="LTR"** (left-to-right) respektive **DIR="RTL"** (right-to-left). DIR anges, precis som LANG ovan, antingen som attribut till <HTML> för hela dokumentet, eller till andra koder för enstaka stycken.

Exempel:

...en webbsida på svenska...
<Q LANG="he" DIR="rtl">*...ett citat på hebreiska...*</Q>
...den svenska texten fortsätter...

Webbläsaren kan i regel inte anta textriktningen utifrån språkangivelsen.

Ändringar i texten

Om man arbetar med ett dokument i en grupp kan man vilja att de andra kan se vilka ändringar som man gjort i texten. Detta kan markeras med styrkoderna **<INS>** för de textbitar som man lagt till och **<DEL>** för de bitar man tagit bort. Koderna avslutas med **</INS>** respektive **</DEL>**. En webbläsare kan sedan visa den markerade texten på något avvikande sätt. Tillagd text kan till exempel visas i en annan färg eller med ett annat typsnitt, och borttagen text kan visas överstruken eller inte alls.

Till <INS> och <DEL> kan man ange attributen **CITE="url"** som anger adressen till en sida där man t.ex. förklarar varför ändringen gjordes, och **DATETIME="datum"** som anger datumet då ändringen gjordes. Datum anges enligt ISO 8601. Exempel:

<INS DATETIME="1997-07-22T18:45:00+02:00">
...denna text lades till den 22 juli 1997 kvart i sju svensk sommartid...
</INS>

<INS> och <DEL> stöds inte av Netscape version 4.

Mer information

W3C (specifikationer till HTML och CCS1)
http://www.w3.org/
Web Review Style Sheets Reference Guide
http://www.webreview.com/guides/style/
Dokumentation till webbservern Apache
http://www.apache.org/
Dokumentation till webbservern NSCA HTTPd
http://hoohoo.ncsa.uiuc.edu/
Dokumentation till webbservern CERN HTTPd
http://www.w3.org/pub/WWW/Daemon/
Netscape Communications Corporation
http://home.netscape.com/
Microsoft Internet Explorer
http://www.microsoft.com/ie/
Tucows (en stor samling av internetprogram för Windows, Macintosh och OS/2)
http://www.tucows.com/
Mag's Big List of HTML Editors
http://sdg.ncsa.uiuc.edu/~mag/work/HTMLEditors/
Pagespinner (HTML-editor för Macintosh)
http://www.algonet.se/~optima/pagespinner.html
JoyHTML (HTML-editor för Windows)
http://www.abc.se/~m8974/joyframe.htm
HomeSite (HTML-editor för Windows)
http://www.allaire.com/
HotDog (HTML-editor för Windows)
http://www.sausage.com/
A Beginner's Guide to HTML by NCSA
http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html
A Beginner's Guide to URLs by NCSA
http://www.ncsa.uiuc.edu/demoweb/url-primer.html
HTML Help
http://www.htmlhelp.com/
Superstars of HTML av Henrik Liljekvist
http://www.henrik.com/superstars/
HTML Guiden av Eva Pepel
http://www.algonet.se/~eva/ref/
Composing Good HTML
http://www.cs.cmu.edu/~tilt/cgh/
HTML Hell page (the good, the bad and the ugly in web design)
http://www.ccil.org/~esr/html-hell.html
The Alertbox: Current Issues in Web Usability
http://www.useit.com/alertbox/

Web pages that suck – Learn good design by looking at bad design
http://www.webpagesthatsuck.com/
The Pinnacle Systems Style Sheet
http://www.pinnaclesys.com/pinnacle/style/index.html
Bandwidth Conservation Society (how to make your web pages display faster)
http://www.infohiway.com/faster/
Optimizing Web Graphics
http://webreference.com/dev/graphics/palette.html
The 216 colors of Netscape which do not dither by Bob Cunningham
http://www.connect.hawaii.com/hc/webmasters/Netscape.colors.html
The Pixel Foundry (tips and tricks with Adobe Photoshop)
http://the-tech.mit.edu/KPT/
GIF89a-based Animation for the WWW by Royal Frazier
http://members.aol.com/royalef/gifanim.htm
Forms tutorial by NCSA
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html
Common Gateway Interface by NSCA
http://hoohoo.ncsa.uiuc.edu/cgi/
The CGI Resource Index
http://www.cgi-resources.com/
Matt's Script Archive
http://www.worldwidemart.com/scripts/
Frequently asked questions about CGI
http://www.htmlhelp.com/faq/cgifaq.html
Cookies
http://home.netscape.com/newsref/std/cookie_spec.html
JavaScript på svenska
http://monkeytoys.com/javascript/
The WWW Security FAQ
http://www.genome.wi.mit.edu/WWW/faqs/www-security-faq.html