HSF
HØGSKULEN I
SOGN OG FJORDANE

NOTAT

På baksiden – Innføring i HTML

Jarle Håvik

N NR 1/2003 AVDELING FOR ØKONOMI OG SPRÅK

På baksiden – Innføring i HTML

Jarle Håvik

N NR 1/2003 AVDELING FOR ØKONOMI OG SPRÅK

HSF
HØGSKULEN
I SOGN OG FJORDANE

NOTAT

Postboks 133, 5801 SOGNDAL telefon 57676000 telefaks 57676100

TITTEL	**NOTATNR.**	**DATO**
På baksiden – en innføring i HTML	1/2003	14. 01. 2003
PROSJEKTTITTEL	**TILGJENGE**	**TAL SIDER**
	Open	93
FORFATTAR	**PROSJEKTLEIAR/-ANSVARLEG**	
Jarle Håvik	Jarle Håvik	
OPPDRAGSGJEVAR	**EMNEORD**	
	HTML, vevsider, lenkje, tabell, rammer, skjema	

SAMANDRAG/SUMMARY

Dette notatet gir en innføring i HTML og hva som skjuler seg bak en vevside. Notatet følger utviklingen fra de enkle html-taggene og frem til å kunne lage en mer fullstendig nettsted. Det gir også en enkel innføring i utforming av skjemaer til bruk for tilbakemelding.

PRIS	**ISSN**	**ANSVARLEG SIGNATUR**
Kr 100,-	**0806-1696**	**Kjell Henden (dekan)**

På baksiden
– en innføring i HTML.

Notatet er laget til bruk i kurs hvor studentene skal får en kort innføring i hva som skjuler seg bak en vevside. Konkret er kursdokumentasjonen tenkt brukt i kursene DA610 - Databaser og datakommunikasjon og BD640 - den elektroniske marknadsplassen.

Notatet har som utgangspunkt at leseren ikke har spesielle kunnskaper om temaet fra før. Vi går veien fra de helt enkle html-taggene og frem til å kunne utforme et eget tilbakemeldingsskjema.

All skriving av html-kode skjer i enkle tekstbehandlere. Dette fordi en på den måten kan konsentrere seg helt og fullt om kodingen, og ikke trekke inn forstyrrende elementer.

Den beste måten å tilegne seg stoffet er å ta utgangspunkt i eksempelskriftene og prøve seg frem med disse. Det er laget oppgaver som finnes bakerst i notatet. Den siste oppgaven inngår som et arbeidskrav for studenter som benytter notatet i kurs på Høgskulen i Sogn og Fjordane.

Det finnes en rekke kilder og ressurs sider på nettet. Tanken bak dette notatet er at det skal kunne brukes som et kort innføringskurs for studenter i kurs hvor internett har en sentral plass. Det er ikke meningen at leseren skal bli ekspert i HTML etter å ha gått gjennom notatet. Snarere er det et mål at han/hun ser at det på baksiden av ethvert html-dokument ligger klare og tildels enkle koder - det å avmystifisere ideen om at å utforme vevsider krever langvarige studier var en av beveg grunnene til å forfatte dette notatet.

Foss, den 14. jan. 2003

_____

Jarle Håvik

Innholdsfortegnelse

1. Introduksjon til HTML

1.1 Mål for kapittelet

Etter å ha lest gjennom dette kapittelet, skal du:

- ha fått en innføring i World Wide Web og HTML
- kjenne eksempler på programmer man kan bruke for å lage web-sider og kunne se på dem
- kjenne nyttige steder å besøke på World Wide Web for å lære mer eller få svar på spørsmål

1.2 Historien bak utviklingen av www og HTML

1.2.1 World Wide Web - kort historikk

World Wide Web (forkortes WWW[i] og kalles ofte "verdensveven" på norsk) har sitt utspring fra forskningssenteret CERN[ii] i Sveits. World Wide Web kan vi føre tilbake til en enkelt person, til engelskmannen Tim Berners-Lee (1955). I 1980 hadde han et seks måneders engasjement ved partikkelfysikksenteret CERN i Geneve, Sveits. I denne periode skrev han for eget private behov, sitt første program for lagring og distribusjon av informasjon. Programmet kalte han ”Enquire” og det kom til å danne basis for fremtidens utvikling av World Wide Web, til tross for at det aldri er blitt publisert.

I 1984 var Tim Berners-Lee tilbake ved CERN. I 1989, fortsatt ved CERN, basert på sitt tidligere arbeid med ”Enquire”, foreslo Tim Berners-Lee et globalt hypertekst prosjekt, et prosjekt for utvikling av et system som ga folk muligheten til kunne samarbeide ved å kombinere deres kunnskaper i en global Web av hypertekst dokumenter. Som en følge av denne ideen, skrev han så den første World Wide Web server[iii] som han ga navnet ”httpd”.

Samtidig med dette skrev han også den første klient browser[iv](heretter brukes det norske navnet nettleser) som han ga navnet World Wide Web (www). Arbeidet med disse startet i oktober 1990 og kom i bruk på Internett sommeren 1991. Dermed var grunnlaget etablert for en utvikling av World Wide Web og Internett, som selv Tim Berners-Lee og kollegene hans ved CERN, neppe hadde forestilt seg. Allerede 1992 var folk utenfor CERN også i gang med utvikling av Web-sider, og Tim Berners-Lee’s initiale spesifikasjoner av HTML, URL og HTTP ble mer og mer forfinet og diskutert i stadig større kretser ettersom Web teknologien fikk stadig større utbredelse. Denne forbedring av HTML, URL og HTTP pågår stadig, men grunnleggeren av alt dette var Tim Berners-Lee.

Web-sider blir skrevet i HTML - et enkelt script språk. Dessverre støtter ikke alle nettlesere de samme tingene. En list over hva d. Derfor bør man ha en viss oversikt over hva av de viktigste ting som de forskjellige nettlesere henholdsvis støtter og ikke støtter. Dermed kan man lage Web sidene slik at flest mulig får tilgang til ens Web sider.

1.2.2 World Wide Web - mediet

En av grunnene til at WWW er blitt så populært, er muligheten til å "lenke" dokumenter sammen. Det vil si å ha en "peker" i et dokument som brukeren kan aktivisere og på den måten få tilgang til et annet dokument. Man kan enkelt gi leserne mulighet til selv å lese dokumenter man siterer fra. I tillegg kan man gi leserne en kort oversikt over andre relevante ressurser som finnes på nettet.

WWW gir ikke bare muligheten til å publisere skriftlig informasjon, men også lyd, bilde og programvare. Mangel på tilgjengelig båndbredde[v] begrenser omfanget av lyd- og filmklipp som legges ut, men etter hvert som båndbredden øker, vil også dette bli mer tilgjengelig. Et eksempel er P4 - Radio hele Norge, som legger ut programmet "17.30" på nettet med både lyd og bilde. Det er imidlertid kun tilgjengelig for brukere med ISDN og båndbredden ISDN gir.

WWW har ikke bare fordeler, men også en god del ulemper. En av grunntankene bak WWW er tilgjengelighet for alle. Denne grunntanken mener mange har forsvunnet mer og mer etter hvert som kommersielle interesser har gjort sitt inntog. Behovet for å tiltrekke seg lesere har gjort at man har tatt i bruk de siste nyvinninger og kombinert dette med fancy layout. Resultatet er etter manges mening at WWW har blitt Internettets[vi] versjon av neonreklame-skilt; de er fine å se på, men det er vanskelig å finne fram til nyttig informasjon der.

1.2.3 HTML (HyperText Markup Language)

For å forklare hva HTML er kan vi ta utgangangspunkt i følgende definisjon hentet fra en FAQ[vii] (Frequently Asked Questions):

> *HTML er et kunstig språk beregnet på strukturering av informasjon i såkalt 'hypertext[viii]'-dokumenter. Stikk i strid med vanlig oppfatning er HTML ikke en måte å beskrive* ***utseendet*** *på hjemmesider.*

HTML er altså et kunstig språk, oppfunnet for å dekke behovet for å strukturere informasjon i et dokument. Når man lager HTML-dokumenter (web-sider), merker man teksten ut fra den strukturen man ønsker at dokumentet skal ha. Med dette mener vi at man legger inn informasjon i dokumentet om hva som er overskrift, avsnitt, sitat osv. En web-side skal derfor i utgangspunktet ikke si noe om hvordan tekstelementene skal se ut, men i stedet beskrive hva de er (avsnitt, overskrift, sitat osv).

Du synes kanskje det er vanskelig å tenke slik. Vanligvis pleier man å konsentrere seg om hvordan ting skal se ut og hvor det skal plasseres. Det er dette som kalles layout, eller sideutforming på godt norsk. HTML ble ikke laget for å være et layout-språk, og derfor er det vanskelig å drive med layout i HTML. I stedet tenker man *struktur*.

Når du tenker struktur, tar du i bruk allerede definerte stiler og markerer tekst med disse (avsnitt, overskrifter, sitater osv). Det er det samme som du lærer om i kurs i tekstbehandling. Der markeres tekst med stiler, og disse stilene er så definert med tanke på utseende og plassering. I HTML er det mange likheter med tekstbehandling. Den store forskjellen er at stilene i HTML er definert på forhånd. Det vil si at du har et sett med stiler du kan benytte deg av. Dessuten vil du ha liten eller ingen mulighet til å påvirket utseendet til disse stilene. I HTML er det i stedet nettleseren som bestemmer utseendet på dokumentet.

HMTL er ikke ment å brukes til å lage layout på web-sider. Det er imidlertid mulig å lage noe layout i HTML, men det er mange begrensninger. Siden du ikke vet hvilket oppsett nettleseren til dine besøkende har, sitter du uten garanti for at layouten blir bra. Det er fordi forskjellige nettlesere og operativsystem har ulike skriftsnitt tilgjengelig, ulik størrelse på disse, ulik størrelse på nettleser-vinduet osv. Du har derfor aldri noen garanti om hvordan web-siden vil se ut for en besøkende.

1.3 Programvare

1.3.1 Programvare for å lage web-sider

Programvare som brukes for å lage web-sider, kan deles i to grupper:

I. WYSINWOG (What You See Is Not What Others Get)

II. Teksteditorer [ix]

Den første gruppen har sitt utspring i utrykket "WYSIWYG - What You See Is What You Get". WYSIWYG er et begrep som raskt ble populært da tekstbehandlere for Microsoft Windows 3.0 kom, og man så et ark på skjermen. Tekstbehandlere med WYSIWYG skulle sørge for at det arket du så på skjermen, så likt ut når det ble skrevet ut på en skriver.

På WWW gjelder ikke dette utrykket nettopp fordi man ikke har noen garanti for at besøkende ser et dokument som er likt det man så når man utformet det.

Noen programvareprodusenter har imidlertid ønsket å gi inntrykk av at man på WWW kan oppnå WYSIWYG, og laget programmer hvor man redigerer en web-side på samme måte som man gjør i en tekstbehandler. Dermed oppstod uttrykket "WYSINWOG - What You See Is Not What Others Get", som impliserer at det du ser på din skjerm ikke nødvendigvis er det samme de besøkende ser på sine skjermer.

Dette betyr ikke at du ikke kan bruke WYSINWOG-programvare. Du må bare være klar over at det er enkelte begrensninger forbundet med bruken av dem. Fordelene med disse programmene er at du ikke trenger å sette deg inn i HTML. I stedet kommer du raskt i gang, og får snart et resultat du selv er fornøyd med. Ulempen er at du ikke får forståelse for hva som ligger "under skallet" og dermed behandler dokumentene som om de var skrevet med en tekstbehandler. Med "under skallet" mener vi hvordan en web-side ser ut dersom du titter på den med et program som viser filen som rå tekst (*ikke* en tekstbehandler). Da vil du se kildekoden for dokumentet, og web-siden vil ikke se ut slik den gjør i en nettleser.

Eksempler på WYSINWOG-programvare som er tilgjengelig er:

- Microsoft Frontpage
- Netscape Composer
- Sausage Software HotDog

Microsoft Frontpage er kommersielt tilgjengelig og kan kjøpes i butikken. Netscape Composer er gratis og følger med Netscapes programpakke "Communicator". Sausage Softwares HotDog er shareware, det vil si at du kan laste det ned via Internett og få en prøvetid hvor du kan teste ut programmet. Dersom du vil bruke programmet permanent, må du imidlertid betale for det.

Det finnes mye programvare som er tilgjengelig som shareware. Dersom du ønsker å laste ned et slikt program fra nettet, anbefaler vi deg å undersøke et av de følgende nettstedene:

- http://www.tucows.com/
- http://www.shareware.com/

Den andre gruppen programvare som brukes for å lage websider, er teksteditorene. Her arbeider man med filen "under skallet", det vil si at man redigerer tekstfilene direkte. Dette gir større grad av kontroll over det man lager. Ulempen er at man må sette seg inn i HTML. Derfor vil man sannsynligvis bruke lengre tid før det første dokumentet er ferdig. I tillegg kan det være vanskelig å se for seg det ferdige resultatet, fordi endringene vises i editoren og ikke slik det vil se ut i nettleseren. Eksempler på tekst-editorer er:

- Notisblokken i MS Windows ("Notepad" dersom du har en engelsk versjon av Windows)
- Arachnophilia
- Easy HTML 2.2.5.181
- HTML Webmaster 2.1

Notisblokken følger med Microsoft Windows. Arachnophilia er en tekst-editor for Windows og denne finner du på studentserveren i mappen. For andre tekst-editorer anbefales en tur til http://www.tucows.com/ eller http://shareware.cnet.com/(her må du så skrive inn søkebegrepet HTML-editors for å søke på nettopp HTML-editorer).

I dette kurset vil vi bruke tekst-editorer for å lage websider. Hvilken editor du velger å bruke, er opp til deg. Mengden tekst som vil bli skrevet inn, er ikke så stor at valg av editor spiller noen særlig rolle, men til senere arbeider vil editorer som gir deg mulighet for hurtigtaster, fargelegging og utheving av tekst være en fordel, siden de letter arbeidet. Det vil ikke bli gitt brukerstøtte for editorene, så du er selv ansvarlig for at programmet fungerer som det skal. Med tanke på eksamen, hvor dere kan får konkrete spørsmål om HTML-koding er det kanskje lurt å skrive inn koden i en enkel editor (for eksempel Notepad).

1.3.2 Programvare for å se på web-sider - nettlesere

En nettleser er et program man bruker for å se på web-sider. Egentlig er det feil å benytte ordet nettleser (eller det engelske ordet "browser"); i stedet burde man bruke "user agent[x]". "User agent" er hovedtermen og omfatter nettlesere, søkeroboter[xi] og lignende. For enkelhets skyld vil jeg imidlertid heretter kun snakke om nettlesere.

Hvilke nettlesere som er tilgjengelige, kommer an på hvilket operativsystem du bruker. For Windows er de to mest kjente nettleserne Microsoft Internet Explorer og Netscape Navigator. Men vi har også en sterk og god norsk nettleser fra Opera Software.

Det finnes omtrent 700 forskjellige nettlesere tilgjengelig, så dersom du ønsker å finne en annen nettleser, er det ikke så vanskelig. Det kan være lurt å ha tilgang til flere nettlesere når du skal lage web-sider. Det er nemlig en fordel å se på sidene

med mer enn én nettleser dersom du ønsker å lage web-sider som skal være tilgjengelige for en bred brukermasse. Det kan også være nyttig å bruke eldre versjoner av enkelte nettlesere, fordi de ofte ikke støtter den versjonen av HTML du har benyttet deg av. Dermed kan du raskt finne ut hvorvidt informasjonen på web-siden din fortsatt er lett tilgjengelig (det vil si er såkalt "bakoverkompatibel"). I tillegg er det en god ide å teste siden ut på forskjellige operativsystem.

Lynx er også en nyttig nettleser å ha tilgang til. Den er en tekstbasert nettleser og er tilgjengelig på de fleste Unix-systemer. Når du ser på web-sider med Lynx, får du en god pekepinn på hvordan dokumentet ser ut for besøkende som har en browser med talesyntese (dvs. leser opp teksten på skjermen) eller bruker leselist. Dersom du ikke har tilgang til Unix og Lynx, kan du besøke nettstedet Web Page Backward Compatibility Viewer. Her kan du undersøke hvordan web-sidene du lager ser ut for besøkende med en nettleser som ikke støtter alt det din egen gjør.

1.3.3 Valg av programvare

Hvilke programmer du velger å bruke, avhenger av en rekke faktorer. Noen stikkord er:

- begrensninger i maskinvare
- ønske om et arbeidsmiljø man er vant med
- begrenset tid til å sette seg inn i HTML

På grunn av begrenset økonomi vil tilgjengelig maskinvare være en faktor. Ikke alle har tilstrekkelig maskinkraft til å holde følge med utviklinge

n innen programvare. Resultatet kan være at du ikke kan ha fem nettlesere, siden du bare kan kjøre én på skjermen av gangen, eller at du ønsker å ha en editor som bruker lite ressurser. I tillegg kan harddiskplass sette begrensninger for hvor mye man har plass til.

Andre vil i stedet prioritere å ha et arbeidsmiljø man kjenner. Det er her bruken av tekstbehandleren Microsoft Word kommer inn. Word har en begrenset mulighet til å konvertere dokumenter til HTML. Fordelen med det er at du kan jobbe med et program du kjenner og derfor slipper å sette deg inn i ukjente programmer og funksjoner. I Word arbeider du med dokumentet på vanlig måte, men når du skal lagre det, velger du "Lagre som/Save as" på Fil-menyen, og deretter spesifiserer du filtypen som HTML-dokument. Dokumentet blir da lagret som en web-side. Løsningen medfører en del ulemper. Word er en *tekstbehandler*. Du kan derfor ende opp med å lage web-sider hvor du i utstrakt grad tenker *layout* i stedet for *struktur*, og som tidligere nevnt er ikke HTML et layout-språk.

I tillegg bør du være klar over at HTML-sidene Word lager, ikke er i nærheten av å bruke korrekt HTML. Dette kurset tar utgangspunkt i at HTML-spesifikasjonene W3C utgir, er å anse som "korrekt HTML". Korrekt HTML er et greit utgangspunkt for å lage web-sider som er tilgjengelige med flere nettlesere. Til slutt i denne studieenheten vil du lære hvordan du sjekker etter feil i web-sidene ved hjelp av en validator. Dersom du bruker Word til å lage web-sidene dine, vil de aldri være feilfrie.

Tid er noe man aldri får nok av, og mange skulle sikkert ønske at døgnet hadde flere timer slik at man rakk alt man ville gjøre. Ved å bruke dette argumentet kan

man velge bort å sette seg inn i HTML og dets finurligheter, og i stedet satse på en enklere løsning. Et WYSINWOG-program vil derfor være den enkleste løsningen.

1.4 Nyttige steder å besøke på WWW

I dette avsnittet vil vi gi deg adressene til noen ressurser som kan gi svar på spørsmål relatert til HTML og WWW. Det kan derfor være nyttig for deg å gå tilbake hit når det oppstår problemer som du trenger hjelp til å løse. Dersom du i stedet har problemer med programvare, henviser vi deg til brukerstøtte for det produktet du har problemer med.

NCSA (at UIUC) Beginner's Guide to HTML

Dette er et innføringskurs i HTML utgitt av The National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. Kurset er på engelsk og tar for seg det grunnleggende i HTML. Det kan være nyttig fordi du får andre forklaringer på de tingene vi tar opp i dette kurset.

URL:http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html.

W3C - The World Wide Web Consortium

World Wide Web Consortium (W3C) tar seg av alt rundt HTML. Dokumentene de utgir er ikke standarder, men i stedet klassifisert som "recommendations" (anbefalinger). Til tross for dette brukes dokumentene omtrent som om de var standarder (og vil heretter i kurset bli brukt som om de var det). Dette er altså stedet å besøke for å finne spesifikasjoner på HTML og andre web-relaterte standarder. I tillegg inneholder W3Cs web-sider en validator som brukes til å sjekke at web-sider ikke inneholder feil i HTML-kodene.

URL: http://www.w3.org/

HTML Help by The Web Design Group

Her vil du finne svar på mange spørsmål angående HTML. Listen over ressurser inneholder ting for HTML-skribenter, arkiv over FAQ'er (Frequently Asked Questions), verktøy som hjelper deg i arbeidet og lenker til *mange* andre nyttige ressurser.

URL: http://www.htmlhelp.com/

I Classfronter vil du finne lenker til disse og flere aktuelle nettsteder. Du vil også her finne henvisninger til mapper på studentserveren hvor du kan hente ned annet aktuelt materiell.

---

[i]Noter til kapittel 1:

[i] World Wide Web (WWW)

World Wide Web var i utgangspunktet et medium som ga mulighet til å aksessere dokumenter ved hjelp av TCP/IP og HTTP. TCP/IP er protokollen som har gjort

Internett mulig, HTTP (HyperText Transfer Protocol) er en overføringsprotokoll som ble skapt for å gjøre WWW mulig.

Siden har WWW blitt utvidet slik at man kan bruke andre protokoller (for eksempel FTP (File Transfer Protocol)).

Federal Standard 1037C : "Glossary of Telecommunication Terms" definerer WWW slik:

World Wide Web (WWW): An international, virtual-network-based information service composed of Internet host computers that provide on-line information in a specific hypertext format. *Note 1:* WWW servers provide hypertext metalanguage (HTML) formatted documents using the hypertext transfer protocol (HTTP). *Note 2:* Information on the WWW is accessed with a hypertext browser such as Mosaic, Viola, or Lynx. *Note 3:* No hierarchy exists in the WWW, and the same information may be found by many different approaches.

[ii] CERN

Forskningsinstitusjon på grensen mellom Nederland, Belgia og Luxemburg.

[iii] Server (på norsk: tjener)

Også kalt "vertsmaskin", en maskin som gir tilgang til tjenester for klienter (brukere). Hvilke tjenester det gis tilgang til er avhengig av konfigurasjonen til serveren.

[iv] **Browser** (norsk: Nettleser)

Programvare som gir en bruker tilgang til World Wide Web

[v] Båndbredde

Tilgjengelig kapasitet for overføring av data. Høyere/større båndbredde betyr raskere overføringer. Når man bruker Internett er det viktig å huske at båndbredde ikke betyr hastigheten du kobler deg opp mot Internett-leverandøren, ettersom muligheten for å hente filer fra for eksempel USA kan være bestemt utfra hvor mye båndbredde som er tilgjengelig på kabelen over Atlanterhavet.

[vi] Internett

Et verdensomspennende nettverk av datamaskiner. Internett er basert på to protokoller, TCP og IP, og på disse ligger et stort antall tjenester tilgjengelig. Eksempler er World Wide Web, elektronisk post, filoverføring og nyhetsgrupper.

[vii] FAQ/OBS/OSS/SOS

En samling med ofte stilte spørsmål, og svarene på disse. Slike dokumenter er meget utbredt på Usenet, og de summerer opp spørsmål som stilles ofte. Nykommere kan da henvises dit slik at man unngår å bruke ressurser på å diskutere problemet en gang til. De forskjellige forkortelsene står for:
FAQ - Frequently Asked Questions,
OBS - Ofte Besvarte Spørsmål,
OSS - Ofte Stilte Spørsmål
SOS - Spørsmål Og Svar.
Hva som bør brukes på norsk er man ikke enig om.

[viii] Hypertekst

I elektronisk sammenheng betyr hypertekst at teksten er lagret slik at mens man leser et dokument, kan man følge en referanse (link) til et annet dokument og lese det før man returnerer til den første artikkelen eller går videre til noe annet (ved å følge en ny referanse, eller se på et nytt dokument).

[ix] Tekst-editor

Et program for å redigere tekst, gjerne i flere forskjellige formater.

[x] User Agent

Fellesbetegnelse for programmer som gir brukere tilgang til web-sider. Betegnelsen inneholder derfor alle typer nettlesere, søkeroboter og enhver annen måte å få tilgang til web-sider på.

[xi] Søkerobot

Et program som benyttes av mange kjente steder på World Wide Web hvor man kan søke etter informasjon (for eks. Alta Vista). Programmet settes til å sjekke web-sider og følger på egenhånd linker på sidene, slik at et stort antall web-sider raskt kan gjøres søkbare.

2 Grunnleggende elementer i HTML

Vi skal nå gå i gang med å lage en enkel web-side. Før vi skriver selve siden, er det imidlertid viktig at du har kjennskap til en del begreper for å unngå misforståelser.

2.1 Mål for kapittelet

Etter å ha lest gjennom dette kapittelet, skal du:

- kunne lage en enkel web-side
- skjønne forskjellen mellom blokk- og tekstnivå
- se sammenhengen mellom tagg og attributt
- sjekke web-siden for feil ved hjelp av en validator

2.1 Noen viktige begreper innen HTML

2.1.1 Element

Tidligere skrev vi at HTML brukes til å beskrive strukturen i et dokument. Til dette bruker vi forskjellige *elementer*. Eksempler på elementer er avsnitt, tabeller, overskrifter og lister. Et element kan derfor sees på som en byggekloss, og web-siden er byggverket vi reiser med byggeklossene. Et element kan inneholde tekst, andre elementer eller begge deler.

2.1.2 Tagg

Et element bygges opp av en eller flere *tagger*. En tagg starter med '<' (mindre-enn) og slutter med '>' (større-enn). Mellom disse tegnene vil du finne tekst som beskriver hva taggen gjør, og det kaller vi *tagg-navnet*.

Et element består som oftest av to tagger, en *start-tagg* og en *slutt-tagg*. Forskjellen mellom disse er at det i slutt-taggen settes inn en '/' (skråstrek) før tagg-navnet. Et eksempel på dette er tabellelementet som har start-taggen `<TABLE>` og slutt-taggen `</TABLE>`.

Enkelte elementer består bare av en start-tagg. For eksempel lager vi et mykt linjeskift ved å bruke taggen `<BR>`.

Når du skriver inn tagger i dokumenter, spiller det ingen rolle om du bruker store eller små bokstaver, men det lønner seg å være konsekvent. For eksempel kan start-taggen for tabeller skrives `<TABLE>`, `<table>` eller med en blanding av store og små bokstaver f.eks. `<tAbLe>`. I dette kurset vil alle tagger skrives med store bokstaver. På den måten er det lettere å skille mellom tagger og omkringliggende tekst. I kildekoden til dette kurset er imidlertid alle taggene skrevet med små bokstaver. Dette fordi dette har blitt en standard i XML, som er ment å erstatte av HTML.

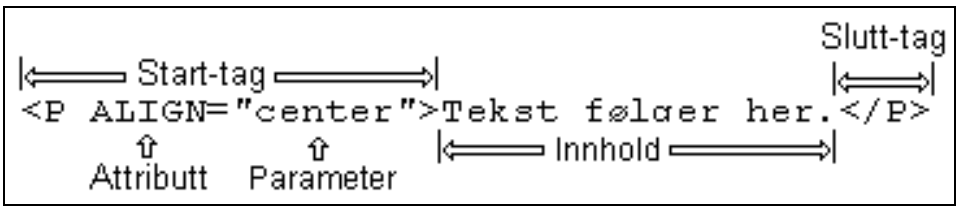
2.1.3 Attributter

Et element består som nevnt av to tagger, en start-tagg og en slutt-tagg. Mange av elementene har den egenskapen at start-taggen kan utvides ved å legge inn *attributter*. Et eksempel er lenker, som du skal lære om senere i dette kapittelet. Lenker lages ved å bruke anker-taggen `<A>` (fra det engelske Anchor) og sette den sammen med attributtet `HREF=""`. Den ferdige start-taggen blir da `<A HREF="">`.

En start-tagg vil alltid starte med en <, inneholde én tagg med ett eller flere attributter, og så avsluttes med >. Husk imidlertid at slutt-tagger aldri inneholder mer enn én tagg (m.a.o ingen slutt attributter!).

2.1.4 Tagg, attributt, parameter og innhold

Det er fire begreper det er viktig at du kjenner forskjellen på før du går videre: tagg, attributt, parameter og innhold. Uttrykkene "tagg" og "attributt" er forklart i de to foregående avsnittene, her skal vi ta for oss de to andre. Som eksempel skal vi bruke koden for et avsnitt med tekst som skal sentreres. Figur 1.1 viser koden til dette avsnittet med forklaringer:



Figur 2.1.: Sentrert avsnitt med forklaringer

Avsnittet er i utgangspunktet begrenset av `<P>` og `</P>`, start- og slutt-taggene for avsnitts-elementet. Innholdet i avsnittet, det vil si teksten som blir markert som et avsnitt er "Tekst følger her."

Til slutt har vi attributtet `ALIGN=""` som bestemmer justeringen av avsnittet. Attributtet trenger et parameter for å bestemme justeringen, og vi setter inn `"center"` fordi vi ønsker et sentrert avsnitt (mer om dette i kapittel 3). Vi setter attributtet inn i start-taggen for avsnittet, og får da den nye start-taggen slik den er vist på figuren: `<P ALIGN="center">`.

Det er viktig at du legger merke til terminologien som er brukt på figuren og hva de forskjellige uttrykkene betyr. Senere vil vi forklarer hvordan de forskjellige elementene fungerer og bygges opp med attributter.

2.1.5 Blokknivå og tekstnivå

Når du jobber med HTML, er det viktig at du vet forskjellen på et **blokknivåelement** og et **tekstnivåelement**. På engelsk kalles de henholdsvis "block level element" og "text level element".

Forskjellen mellom dem ligger i hvorvidt elementet vises med et mellomrom ("paragraph break"/avsnittsbrekk) før og etter i nettleseren. Nettleseren vil sørge for at et blokknivå-element har et mellomrom før og etter seg. Dette mellomrommet er vanligvis en tom linje. Et eksempel på et blokknivåelement er et avsnitt (`<P>`), som sørger for en tom linje før og etter tekstavsnitt.

Et tekstnivåelement vil ikke ha noe avsnittsbrekk, men i stedet flyte inn i teksten. Elementer på tekstnivå blir derfor brukt til endring av tekst, f.eks. **fet skrift**, understreking og lenker (denne går til VG`s hjemmeside). Som du sikkert la merke til, blir det ikke laget tomme linjer før og etter de tre effektene (fet skrift, understreking og lenker).

2.2 Inndeling av web-siden

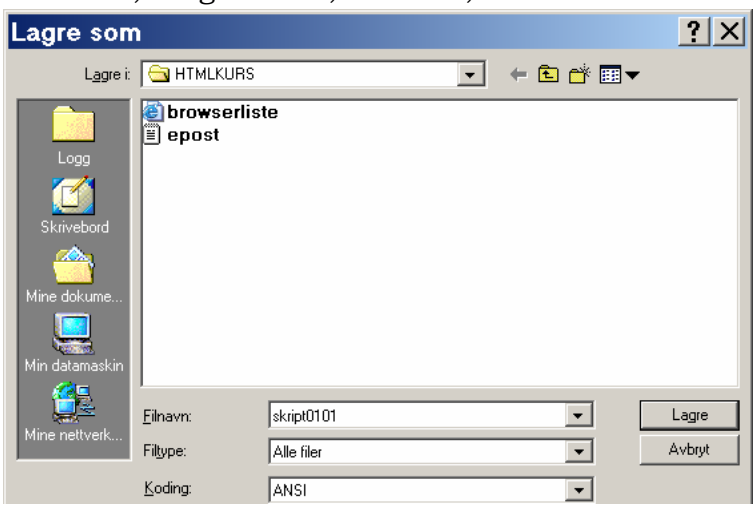
En web-side kan deles inn i tre deler: deklarasjon, dokumentets hode og dokumentets kropp. Funksjonen til de tre delene lar seg enkelt forklare ved å bruke et eksempel. Derfor lager vi en ny web-side og illustrerer dem.

Når du skal lage en ny web-side, er det lurt å lage en egen mappe/katalog til å lagre filene i. Hvilket navn du gir denne mappen/katalogen er opp til deg, men det kan være lurt å opprette en mappe som heter HTMLKURS. I denne mappen kan du så lage undermapper for hvert av kapitlene i dette notatet. Dette hjelper deg til å huske hva du har lagret i mappen/katalogen, samtidig som det gjøre det lettere å overføre sidene til en web-server.

Neste skritt er å starte tekst-editoren. Dersom du bruker Windows, og har valgt å bruke Notisblokken, finner du den under "Start, Programmer, Tilbehør, Notisblokk".

Når du bruker editoren, bør du huske å lagre det du skriver jevnlig, slik at det ikke går tapt dersom noe skulle gå galt. Hvis du har valgt å bruke Notisblokken i Windows, vil du kunne lagre ved å velge "Fil" på rullegardinmenyen, og deretter velge "Lagre som...".

Pass også på at filtypen er satt til alle filer.



Figur 2.2: Lagre HTML-skript i Notepad.

Lag en ny fil og lagre den som "skript0101.html". Hvis du ikke lagrer filen med fil-endelse ".html" vil du ikke få den opp når du ber programmer om å liste HTML-filer, siden de kjennetegnes med endelsen ".htm" eller ".html". Når du har opprettet en ny fil, skriver du inn følgende:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<html>

  <head>

  </head>

  <body>

  </body>

</html>
```

Skript0201.html: HTML-skjelettet

2.2.1 Deklarasjon

Den første linjen, som begynner med `<!DOCTYPE...`, er "deklarasjonen". Denne linjen er ikke et element, men en SGML-deklarasjon. Informasjonen på denne linjen forteller deg at dette dokumentet benytter HTML i henhold til standarden for versjon 3.2 utgitt av World Wide Web Consortium (W3C). Filen din må inneholde en slik deklarasjon når du senere skal kontrollere den for feil i en validator. Da forteller deklarasjonen validatoren hvilken versjon av HTML den skal kontrollere. Deklarasjonen skal også fortelle nettleseren til besøkende på web-siden din hvilken versjon av HTML som er benyttet. I tillegg er deklarasjonen en huskelapp som forteller deg hvilken versjon *du* skal forholde deg til.

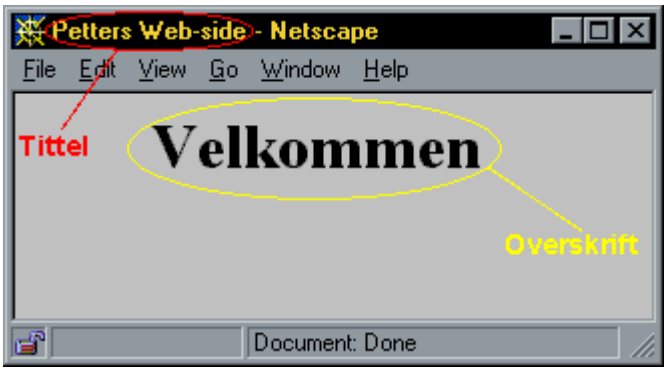
Spesifikasjonene for de versjonene som utgis av W3C, kan du finne på W3Cs web-sider. Siste utgitte standard på det tidspunkt kurset ble skrevet, er versjon 4.0. Vi benytter imdlertid versjon 3.2 i dette kurset, fordi den er mindre og dermed mer oversiktlig for dem som er ukjente med HTML.

2.2.2 Dokumentets hode - <head> og <title>

Dokumentents "hode" begynner med `<head>` og slutter med `</head>`. Mellom disse taggene skal du beskrive hva dokumentet inneholder. Det finnes egne elementer for det, og de fleste er såpass avanserte at vi lar dem vente og tar dem fram igjen i studieenhet 3.

Det er imidlertid i dokumentets hode at du skal legge inn tittelen på dokumentet ditt. Til dette benytter vi tittelelementet, som begynner med `<title>` og slutter med `</title>`. Mellom start- og slutt-taggen skal du skrive inn dokumentets tittel.

Tittelen du legger inn kommer ofte opp på programlinjen til nettleseren din. Den må du ikke forveksle med eventuelle dokumenttitler, som skrives som overskrifter. Figur 2 viser et eksempel, hvor tittelen er "Petters Web-side", mens overskriften er "Velkommen":



Figur 2.3: Tittel og overskrift.

Skriv inn tittelen på din nyopprettede web-side. Tittelen kan for eksempel være "Hallo verden - min web-side". Lag en tom linje mellom `<head>` og `</head>` og skriv inn følgende:

```
<title>Hallo verden - min web-side</title>
```

Dokumentet ditt skal da se slik ut:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Hallo verden - min web-side</TITLE>
</head>
<body>


</body>
</html>
```

Skript0201b.html: Tittellinje

2.2.3 Dokumentets kropp - <body>

Dokumentets kropp er naturligvis det som får mest oppmerksomhet, fordi det er her selve informasjonen på web-siden befinner seg. Du skal derfor lære å kjenne mange forskjellige elementer som brukes i dokumentets kropp. HTML er imidlertid omtrent som en tekstbehandler, 90 % av tiden bruker man 10 % av mulighetene. Derfor skal vi først ta for oss de mest sentrale elementene som du vil få bruk for. Taggen som definerer dokumentets kropp, begynner med `<body>` og avsluttes med `</body>`. Mellom disse kan du sette inn selve teksten på web-siden din.

Du lurer kanskje på hvor `<html>` og `</html>` ble av? Disse to taggene vil alltid være med, men de gjør ikke annet enn å fortelle at innholdet mellom de to taggene er HTML. De to taggene vil derfor bare være med i eksempler der koden for hele dokumentet vises.

2.2.4 Avsnitt - <p>

Web-siden din er for øyeblikket tom. La oss fylle den med et vanlig uttrykk fra programmeringsverdenen: "Hallo verden!" For å hjelpe deg med dette skal vi introdusere det første elementet som hører hjemme i dokumentets kropp. Dette elementet definerer et avsnitt ("paragraph" på engelsk), og startes med start-taggen `<p>` og avsluttes med slutt-taggen `</p>`. Teksten som befinner seg mellom start- og slutt-taggen, utgjør altså et avsnitt. Et avsnitt er definert som et blokknivåelement.

Lag en blank linje mellom `<body>` og `</body>` og skriv inn følgende:

```
<p>Hallo verden!</p>.
```

For oversiktens skyld forlater vi nå dokumentets hode og konsentrerer oss kun om kroppen. Filen du redigerer, må fortsatt ha med deklarasjonen og dokumentets hode, men de delene vil vi utelate i de fleste eksemplene bortsett fra når vi viser fram hele dokumentet.

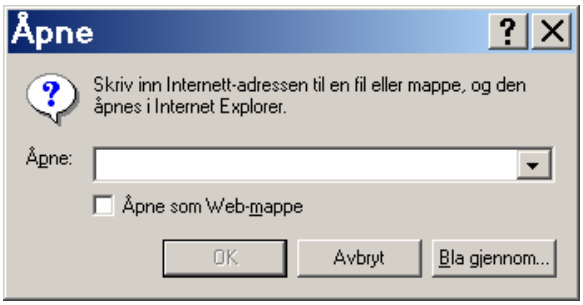
I din fil skal dokumentets kropp nå se slik ut:

```
<body>
<p>Hallo verden!</p>
</body>
```

Skript0201c.html: Hallo Verden

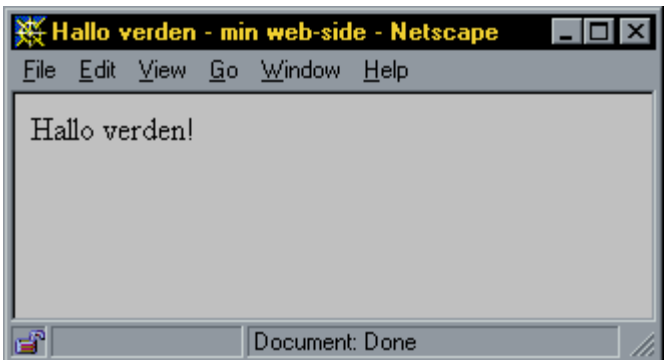
Du har nå laget en web-side, og det er sannelig på tide at vi ser hvordan den ser ut. Det er svært viktig at du husker på å lagre filen før du åpner den i nettleseren din.

Første skritt for å få sett på web-siden din er å starte nettleseren. Deretter ber du nettleseren om å åpne en lokal fil, ettersom dokumentet ditt befinner seg på din harddisk og ikke ute på WWW. I Microsoft Internet Explorer velger du "Fil", deretter "Åpne" og så knappen merket "Bla gjennom...". Når du har valgt filen trykker du på "Ok".



Figur 2.4: Åpne et HTML-dokument

Web-siden du nå får opp, skal se omtrent slik ut:



Figur 2.5: Web-siden, slik den vises i Netscape Navigator.

2.2.5 Overskrift - <h1>

I HTML er det mulig å bruke overskrifter på seks nivåer. Nivå 1 er ment for de mest framtredende overskriftene og blir derfor vektlagt mest av nettleseren. I en grafisk nettleser vil dette gjerne vises med stor og fet skrift. Nivå 6 representerer den minst framtredende overskriften og blir vektlagt minst av nettleseren. Det er viktig å huske på at alle overskriftselementene, uavhengig av nivå, er ment som overskrifter og derfor ikke skal brukes for å få større skrift eller framheve tekst i et avsnitt.

Det kan være vanskelig å tenke seg hva vi mener med den "mest framtredende" overskriften. Du er kanskje vant til å jobbe i en tekstbehandler? Der vil overskrifter ofte være skrevet med stor, fet skrift. HTML er ikke et layout-språk, og derfor er det ikke du som bestemmer hvordan overskriftene skal se ut. I stedet sier du hvilket nivå overskriften skal ha. I en grafisk nettleser, som for eksempel Netscape Navigator, vil overskrifter på nivå 1 (mest framtredende) ha stor, fet skrift, mens

overskrifter på nivå 6 er mindre og ikke så synlige. Layouten vil imidlertid variere i de ulike nettleserne.

Overskriftene på nivåene 1 til 6 vil vises slik i nettleseren Netscape Navigator under Windows 98 med standard oppsett av skrift-typer:

Når en skal jobbe med overskrifter, er det viktig å benytte de forskjellige nivåene flittig og riktig.

Det eksisterer flere filosofier angående nivåer, men de har ett fellestrekk:

- den første overskriften i et dokument skal være på nivå 1, og er dermed hovedoverskriften i dokumentet (gjerne dokumentets tittel).

Figur 2.6: Overskriftsnivåene, slik de vises Netscape Navigator..

Hvordan man bruker nivå under hovedoverskriften, er det delte meninger om, men en god huskeregel kan være denne:

De overskriftene som følger etter hovedoverskriften, og som skal vektlegges mest, får nivå 2. Underoverskrifter under nivå 2 får nivå 3 osv.

Overskrifter lager du ved å benytte start-taggen <h**x**>, hvor '**x**' representerer et tall som viser nivået på overskriften og varierer fra 1 til 6. Slutt-taggen er </h**x**>, hvor 'x' er tallet som viser nivået på overskriften og skal samsvare med tallet i start-taggen. Overskrifter er også et blokknivåelement.

Nå kan du forsøke å sette inn en overskrift på web-siden din. Den bør fortelle besøkende hva siden inneholder. Fordi dette er den første overskriften du setter inn, vil det være den første på siden, og du skal velge overskriftsnivå 1. Du skriver da inn (etter <body>, men før avsnittet du satte inn tidligere):

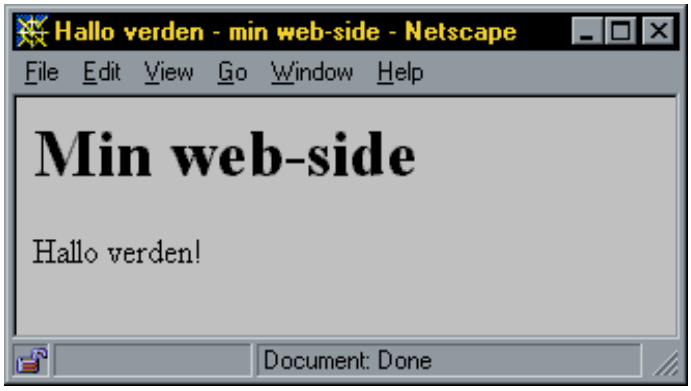
```
<h1>Min web-side</h1>
```

I filen du har i editoren din, skal dokumentets kropp nå se slik ut:

```
<body>
    <h1>Min web-side</h1>
    <p>Hallo verden!</p>
</body>
```

Skript0101d.html: Overskrift

Igjen kan du lagre filen og be nettleseren din om å åpne den. Resultatet skal se omtrent slik ut:



Figur 2.7: Web-siden, slik den vises i Netscape Navigator.

Avsnitt og overskrifter er de to mest brukte elementene i HTML. Dersom du tar for deg et gjennomsnittlig dokument i en tekstbehandler (f.eks. Word), vil du se at mesteparten av dokumentet består av overskrifter og avsnitt med tekst. Tilsvarende gjelder altså også i HTML-dokumenter. Derfor er det ingenting i veien for at du allerede nå kan legge ut filen din på nettet med godt resultat. Vi skal imidlertid gjennomgå noen flere elementer før vi viser deg hvordan du kan gjøre filen din tilgjengelig for andre.

2.2.6 Lenker - <a href=" ">

Siden din er ganske enkel for øyeblikket fordi den inneholder svært lite informasjon. Inntil du får lagt ut mer informasjon, kan du i stedet gi besøkende anledning til å finne fram til en side med større nytteverdi. For å få til det skal vi introdusere taggen man bruker for å lage lenker, og deretter vise deg hvordan du kan lage en lenke til Høgskulen i Sogn og Fjordane (HSF).

En lenke er et tekstnivåelement som starter med start-taggen `<a>` ("a" for det engelske "Anchor") og avsluttes med slutt-taggen `</a>`.

Du skal lage en lenke til HSF på web-siden din. Den kan du skrive inn etter avsnittet med tekst som du la inn til å begynne med ("Hallo verden!"). Du må imidlertid fremdeles holde deg innenfor taggene til dokumentets kropp. Derfor må du skrive inn lenken før `</body>`. Du kan skrive inn følgende for å legge inn lenken:

```
<p> Siden min er under utarbeidelse for tiden, i stedet kan du finne
noe nyttig hos

<a href="http://www.hisf.no">Høgskulen i Sogn og Fjordane</a>.</p>
```

Legg merke til at vi har sagt at denne tekstbiten er et nytt avsnitt siden vi bruker avsnitts-elementet, og legg også merke til hvordan selve lenken flyter inn i teksten.

I dette tilfelle skal lenken gi besøkende mulighet til å forflytte seg til et annet dokument (web-side), og lenke-elementet vil da starte med `<a href="">`. Som du ser, har vi lagt et attributt til start-taggen `<a>`. Attributtet, `href=""`, skal fortelle

nettleseren til den besøkende hvor lenken går. Mellom de to anførselstegnene skal du skrive inn URL'en (adressen) til det dokumentet du vil lenke til.

Du skal lage en lenke til et annet HTML-dokument som har URL'en "http://www.hisf.no". URL'en kan merkes i adressefeltet i nettleseren din, slik at du kan kopiere innholdet og lime det inn i filen du arbeider med.

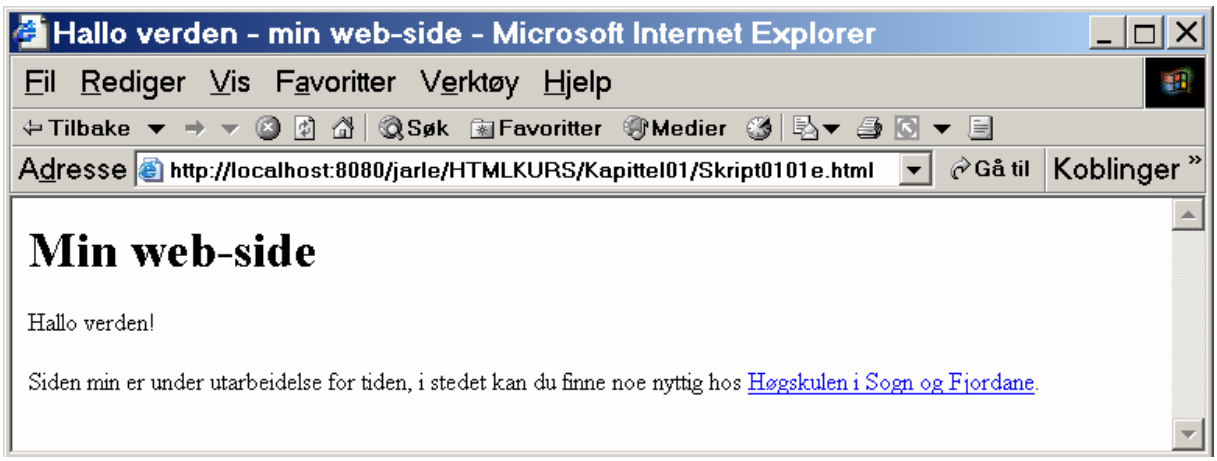
Mellom start- og slutt-taggen må du skrive inn en kort forklaring, slik at leseren forstår hvor lenken fører hen. "Klikk her" er derfor et meget dårlig tekstvalg. Det er fordi enkelte ikke kan "klikke her" og fordi det ikke gir noen forklaring på hvor lenken fører hen. I stedet bør du lete etter gode forklaringer og bruke dem. For eksempel laget du nettopp en lenke til Høgskulen i Sogn og Fjordane, og da bør du fortelle den besøkende dette ved å skrive inn teksten "Høgskulen i Sogn og Fjordane" mellom start- og slutt-taggen. Som du senere vil se, blir denne teksten automatisk merket av nettleseren slik at det går fram at det er en lenke.

Etter at du har lagt inn lenken til høgskulen, skal kroppen til dokumentet se slik ut:

```
<body>
  <h1>Min web-side</h1>
  <p>Hallo verden!</p>
  <p>Siden min er under utarbeidelse for tiden,
    i stedet kan du finne noe nyttig hos
    <a href="http://www.hisf.no">Høgskulen i Sogn og Fjordane</a>.
  </p>
</body>
```

Skript0201e.html: Lenke til ekstern nettside.

Etter at du har lagret endringene i filen, kan du åpne den på nytt i nettleseren din. Da vil den se omtrent slik ut:



Figur 2.8: Web-side med ekstern lenke

Du eller andre som besøker web-siden din, vil kunne følge lenken (som oftest ved å klikke på den mens musepekeren ligger over) og komme til Høgskulen I Sogn og Fjordane, akkurat slik du ønsket.

Med de tre elementene avsnitt, overskrifter og lenker, kan du lage nyttige dokumenter og legge dem ut på WWW. Som vi har nevnt tidligere, er muligheten til å "lenke" dokumenter sammen en av grunnene til at WWW er blitt så populært. Du kan lenke mellom dokumenter *du* lager eller til andre relevante ressurser. Et eksempel på bruk av lenker mellom dokumenter er et større verk, hvor man lenker fra kapittel til kapittel.

2.2.7 Adresse, mykt linjeskift og skillelinje - <adress>,
 og <hr>

I dette avsnittet skal vi ta for oss tre elementer: adresse, skillelinje og mykt linjeskift. Adresseelementet brukes til å markere at et avsnitt beskriver en web-sides opphavsmann. Det er nyttig å fortelle dem som besøker web-siden, hvem som har laget eller er ansvarlig for web-siden de har kommet til, slik at de vet hvem de kan kontakte dersom de lurer på noe.

Adresseelementet har start-taggen `<address>` og slutt-taggen `</address>`. Mellom disse har du mulighet for å legge inn avsnitt og tekst med informasjon om den som har laget eller de som er ansvarlige for web-siden. Adresseelementet er et blokknivåelement.

I ditt dokument skal du legge inn en typisk norsk adresse . Navn, firma, gateadresse eller postboksadresse, postnummer og poststed er vanlige opplysninger å legge inn i adresseelementet. I tillegg er det vanlig å gi besøkende mulighet til å sende e-post til den som har laget eller de som er ansvarlige for web-siden.

En typisk norsk adresse ser slik ut (du bruker selvfølgelig din egen):

Jarle Håvik

Høgskulen i Sogn og Fjordane

Postboks 133

6851 SOGNDAL

Det er vanlig at opplysninger om navn og adresse gjengis slik at hver del kommer på en ny linje, slik du ville ha skrevet det på utsiden av en konvolutt. Det vil si at navnet skal stå på første linje, firma på neste linje, adresse på neste osv., slik det er satt opp i eksemplet over. HTML er imidlertid laget slik at ordmellomrom og linjeskift ikke skal ha betydning for utseendet til dokumentet. Du vil, som i vanlige tekstbehandlere, kunne bruke ordskilletasten til å lage et mellomrom mellom ord. Dersom du prøver å lage to eller flere ordmellomrom, vil det i nettleseren imidlertid bare vises ett. Det samme vil skje ved bruk av linjeskift: ett linjeskift vil "oversettes" av nettleseren til ett ordmellomrom, og to linjeskift vil ha samme effekt som ett linjeskift. Dermed er det ikke noen forskjell på bruk av ordmellomrom og linjeskift når du ser på dokumentet med nettleseren.

For at vi skal få det resultatet vi ønsker, må vi ta i bruk elementet for mykt linjeskift. Et mykt linjeskift skaper et linjeskift i dokumentet, men avslutter ikke avsnittet. I nettleseren vil teksten dermed starte på en ny linje, men den vil fortsatt være en del av avsnittet inntil du avslutter avsnittet. Et mykt linjeskift får du ved å bruke taggen `<br>`, som ikke har noen slutt-tagg. Dette elementet sørger for at du får et linjeskift der taggen er plassert. Elementet er et tekstnivåelement.

Når du skal sette inn informasjon om navn og adresse i et dokument, er det naturlig å gjøre det på slutten av dokumentet. Det er ikke nødvendig å definere et eget avsnitt, siden adresseelementet er på blokknivå. Etter avsnittet med lenken, men før </body>, skriver du inn følgende for å legge inn opplysninger om navn og adresse:

```
<address>

  Jarle Håvik<br>

  Høgskulen i Sogn og Fjordane<br>

  Postboks 133<br>

  6851 SOGNDAL<br>

</Address>
```

Ved å bruke <br> får du hver del av adressen på ny linje. Det er ikke nødvendig å sette inn et linjeskift i filen vi redigerer for å oppnå denne effekten, men det gjør dokumentet ditt mer oversiktlig, og det blir lettere å forestille seg hvordan det vil se ut i nettleseren.

Sammen med adressen er det vanlig å legge en lenke som gjør at besøkende kan sende e-post til den som har laget dokumentet, altså deg. I dokumentet skal du lage en lenke til en vanlig e-post-adresse, for eksempel "jarle.havik@hisf.no" (du kan selvfølgelig bruke din egen). Denne lenken vil imidlertid se litt annerledes ut enn den lenken du laget tidligere til Høgskulen i Sogn og Fjordane. Nedenfor ser du hvordan adresse informasjonen ser ut med e-post-lenken satt inn:

```
<Address>
  Ola Nordmann<br>
  Høgskulen i Sogn og Fjordane<br>
  Postboks 133<br>
  6851 SOGNDAL<br>
  e-post: <a href="mailto:jarle.havik@hisf.no">jarle.havik@hisf.no</a>
</address>
```

Teksten "e-post:" forteller de besøkende at dette er e-post-adressen til den som har laget eller de som er ansvarlige for dokumentet. Deretter følger selve lenken som du raskt gjenkjenner på grunn av elementet <a href="">, som vi gjennomgikk i avsnittet om lenker. I dette tilfellet er imidlertid URL'en litt annerledes. Når du laget en lenke til Høgskulen i Sogn og Fjordane, startet adressen med "http://", men denne gangen starter den med "mailto:". Dette er et felles trekk for lenker som skal gi besøkende tilgang til å sende e-post. Rett etter "mailto:" følger e-post-adressen som posten skal sendes til. I feltet som nettleseren skal vise, bør du skrive inn en tekst som forklarer hvem lenken gir adgang til å sende e-post til. I eksemplet har vi valgt å skrive inn hele e-post-adressen, men det er også vanlig å bruke navnet til forfatteren, institusjonen som er ansvarlig eller lignende.

Før du ser på resultatet med nettleseren, skal du legge inn en skillelinje. Skillelinjen kan du bruke til å skille dokumentets innhold fra adresseinformasjon (slik vi gjør). Linjen brukes også til å dele opp dokumenter, for eksempel for å skille

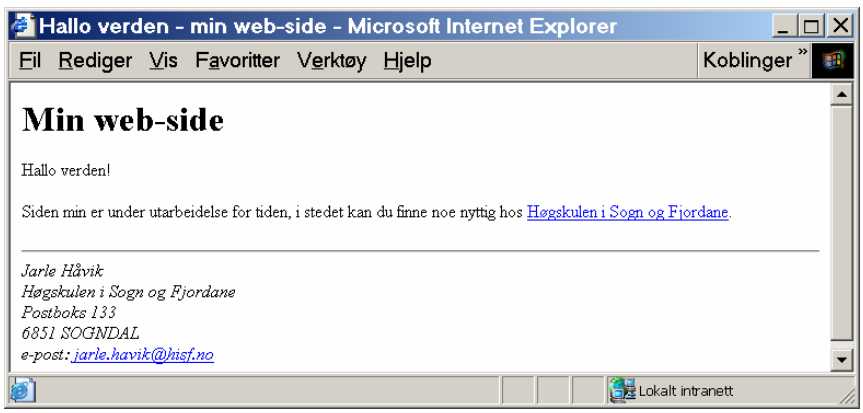
kapitler. Skillelinjen har bare en start-tagg <hr> (fra det engelske "horisontal rule") og avsluttes ikke med en slutt-tagg. Skillelinje er et blokknivåelement.

Du skal sette inn en skillelinje foran adresseinformasjonen, slik at du skiller den fra resten av dokumentet. For å få til dette setter du inn <hr> etter siste avsnitt og før adressen. Dokumentets kropp skal da se slik ut:

```
<body>
  <h1>Min web-side</h1>
  <p>Hallo verden!</p>
  <p>Siden min er under utarbeidelse for tiden,
   i stedet kan du finne noe nyttig hos
  <a href="http://www.hisf.no”> Høgskulen i Sogn og Fjordane </a>.</p>
  <hr>
  <address>
     Jarle Håvik<br>
     Høgskulen i Sogn og Fjordane<br>
     Postboks 133<br>
     6851 SOGNDAL<br>
     e-post: <a href="mailto:jarle.havik@hisf.no">jarle.havik@hisf.no</a>
  </address>
</body>
```

Skript0201f.html: Adresse og e-post.

Nå kan du lagre dokumentet ditt og deretter ta en titt på det i nettleseren. Web-siden vil nå se omtrent slik ut:



Figur 2.9: Web-siden, slik den vises i Microsoft Explorer.

Nå har du laget din første webside. Du har lært hvordan man bruker elementene avsnitt, overskrifter, lenker, adresse, mykt linjeskift og skillelinje. Alle disse elementene kommer du til å bruke mye når du skal lage websider. Det er nå på tide å få kontrollert filen din med W3Cs validator, slik at du vet den er feilfri.

2.3 Validering

For å kontrollere at web-siden din ikke inneholder feil, skal du bruke en validator. Validatoren går gjennom web-siden din og sjekker den mot standarden utgitt av W3C. Dersom du har glemt en tagg, skrevet en tagg feil, eller gjort noe annet som strider mot standarden, vil du få beskjed om det. På den måten er det enkelt å få rettet opp eventuelle feil. Når dokumentet er feilfritt, vil du få en hyggelig beskjed om det, og du kan ta fatt på neste dokument. For å få validert siden hos W3C må den være tilgjengelig på WWW.

For å få validert web-siden din må du først be nettleseren finne fram til "http://validator.w3.org/", som er URL'en til W3Cs validator. Du vil da få opp et skjermbilde som ser omtrent ut som vist på neste side:

Figur 10: Hovedsiden til W3Cs validatortjeneste, slik den vises i Netscape.

Trykk på "PgDn" på tastaturet ditt, slik at du kommer ned til overskriften Validate Documents by URI. Der skriver du inn URL'en til web-siden du vil ha validert. Deretter trykker du på "Submit this URI for validation". Dersom det er feil i dokumentet, vil validatoren nevne disse og gi deg lenker til forklaringer om hva som er feil. Hvis disse forklaringene er uforståelige, ber vi deg i stedet om å ta en titt på web-siden din og kontrollere den mot eksemplet i denne studieenheten.

3. Viktige elementer i HTML

I dette kapittelet skal du lage en ny web-side der du bruker det du lærte i forrige kapittel, og i tillegg utvider med en del nye elementer som kan gjøre web-siden din mer informativ.

3.1 Mål

Etter å ha gjennomgått dette kapitlet, skal du kunne lage en vevside der du bruker følgende elementer:

- Bilder - `<IMG>`
- Manipulering av tekst
- Entiteter for æ, ø og å
- Lister
  - Punktlister - `<UL>`
  - Nummererte lister - `<ol>`
  - Definisjonslister - `<DL>`
- Sitater - `<BLOCKQUOTE>`
- Tabeller - `<TABLE>`

3.2 Bilder - <img src=" ">

Et bilde kan si mer enn ord kan forklare. Ofte er det bedre å illustrere det man vil si med et bilde enn å prøve å forklare det med ord. Historien om de seks blinde mennene som fikk føle på hver sin del av en elefant, er et godt eksempel på det. Etter å ha følt på elefantdelene skulle de diskutere seg fram til hva de hadde kjent på. Hver mann hadde selvfølgelig sin forklaring, men det var ingen av dem som så helheten - nemlig at det var en elefant. I mange tilfeller kan det være bedre å vise et bilde enn å forsøke å forklare med ord.

For å få til dette må du kjenne elementet for å sette inn bilder i HTML. Start-taggen for dette er `<img>`, og elementet har ingen slutt-tagg. Bilder er tekstnivåelementer.

Vi skal bruke bildet av Høgskulen i Sogn og Fjordane for å demonstrere hvordan man legger inn bilder i HTML. Bildet ved siden av ligger i mappen HTMLKURS under ressurser i Classfronter.

Jeg vil under vise hvordan du kan hente et annet bilde ned fra Høgskulen i Sogn og Fjordane sin egen hjemmeside.

Start opp tekst-editoren, og opprett en ny web-side. Til å begynne med skal den se ut som dokumentet til den første web-siden du laget i kapittel 2 (`skript0301.html`), men uten noe innhold i dokumentets kropp.

Du bør forandre tittelen i tittelelementet, slik at det går fram at det ikke er den samme web-siden. For eksempel kan du nå opprette en ny mappe for alle skriptene som lages i kapittel 3 (`kapittel03`. Deretter lagrer du filen med navnet `skript0301.html`. Filen du nå skal starte med, skal da se slik ut:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
   <title>Min andre vevside</title>
  </head>

  <body>
  </body>

</html>
```

Skript0301.html: Skjelett til andre web-side.

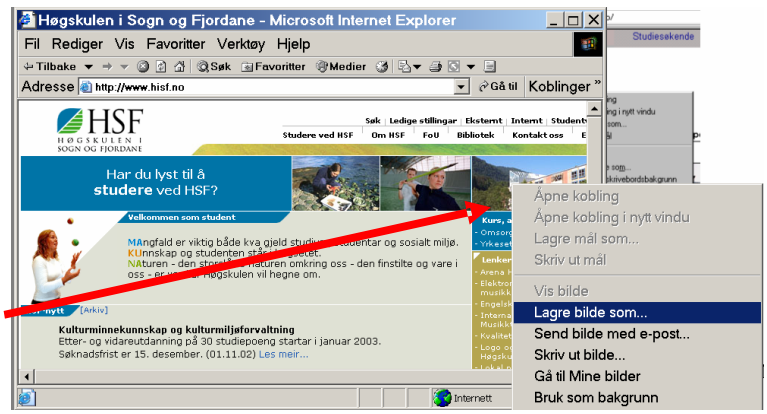
For å sette inn et bilde i dokumentets kropp bruker vi altså taggen `<IMG>`. Denne taggen trenger et attributt for å fungere riktig. For å gjøre det mest mulig forståelig til å begynne med, starter vi med den enkleste versjonen av taggen. Den forteller at det er et bilde og hvor bildefilen befinner seg:

```
<img src="HSF.jpg">
```

Her har vi brukt `<img>`-taggen og utvidet den med attributtet `src=""`. Dette attributtet brukes til kildehenvisning (av engelsk source), og den forteller nettleseren hvor bildefilen befinner seg. I vårt eksempel vil vi at filen som vi kaller hsf.jpg, skal befinne seg i samme katalog/mappe som "skript0301.html" befinner seg i. For å få til det må du hente ned bildet fra WWW.

Dette gjør du det ved å legge musepekeren over bildet du ønsker å lagre, det vil si bildet vi viste over, og deretter klikke på høyre musetast.

Du får da opp en meny hvor du kan velge " Lagre bilde som... ". I dialogboksen du får opp, finner du fram til katalogen som "andreside.html" ligger i og lagrer bildet som hsf.jpg. I Microsoft Internet Explorer gjør du på samme måte, men i menyen du får opp, velger du "Lagre bilde som ..". Deretter finner du fram til riktig katalog i dialogboksen og lagrer bildet.



Figur 3.1: Hente bilde fra en internettside

I utgangspunktet er dette alt du trenger å kjenne til for å få lagt inn et bilde på en vevside. Derfor kan du nå sette inn bildet på web-siden din. Deretter kan vi bruke bildet som et utgangspunkt for å lære noen mer avanserte elementer. Sett inn et

avsnitt i "andreside.html" og legg bildet inn i dette avsnittet. Mellom `<body>` og `</body>` skriver du følgende:

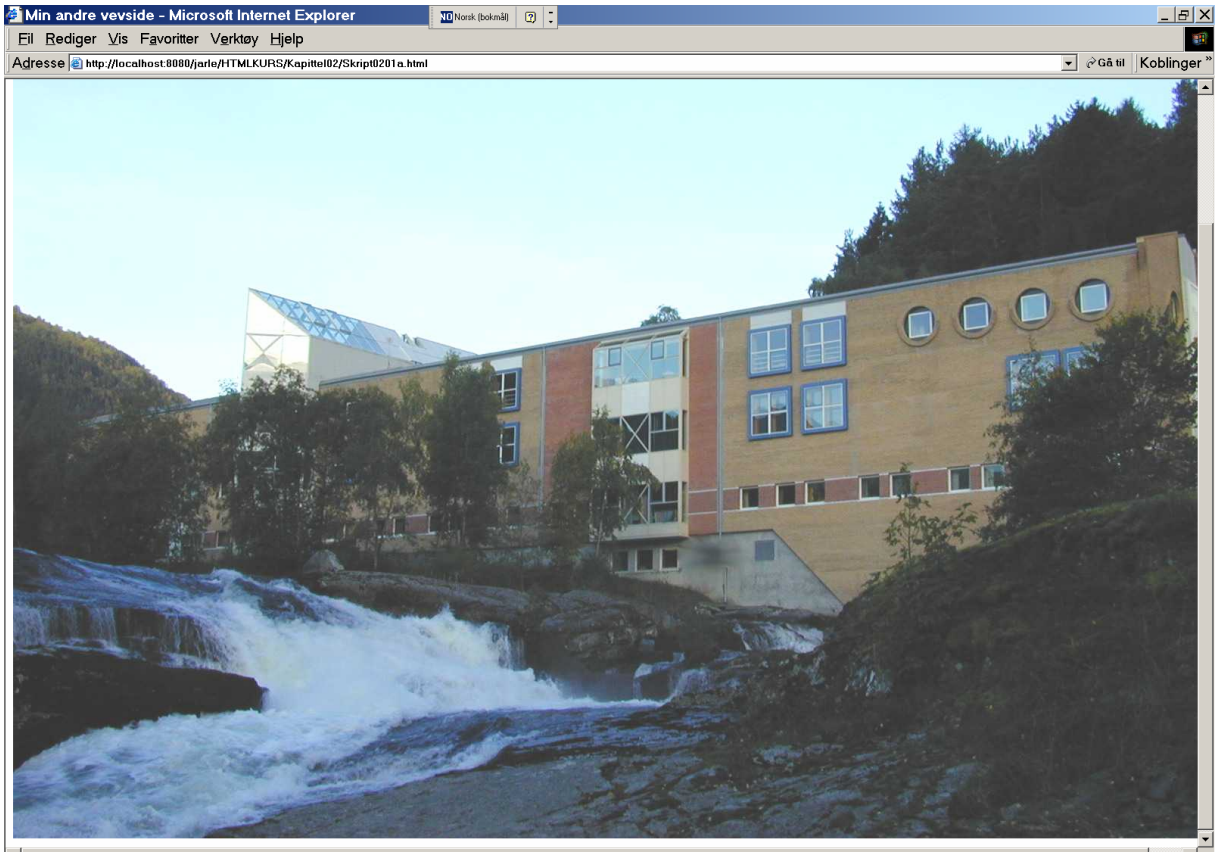
```
<p><img src="HSF.jpg"></p>
```

Hele dokumentet ditt skal da se slik ut:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
   <title>Min andre vevside</title>
  </head>

  <body>

     <p><img src="Hsf.jpg"</p>

  </body>

</html>
```

Skript0301a.html: Sette inn et bilde.

Nå kan du lagre det og deretter åpne det i nettleseren din. Web-siden skal se omtrent slik ut:



Figur 3.2: Web-siden med bilde, slik det vises i nettleseren Microsoft Explorer.

Vi har nå sett hvordan vi kan sette inn bilder på en web-side, men det finnes en del attributter som en kan bruke for å legge inn tilleggseffekter til bilder.

Som du ser ble dette bildet svært stort – det er faktisk ikke plass til hele bildet på de fleste skjermer. For å redusere, eller rettere sagt bestemme hvor stor plass et bilde skal få på en web-side må vi benytte noen tilleggs attributter til `<img>`-taggen. Vi setter bredden og høyden til et bilde ved å legge til attributtene `width=""` og `height=""`. Vi legger nå til disse i img-taggen:

```
<p><img src="HSF.jpg" width="25%" height="25%"></p>
```

Har her valgt å sette bredden og høyden til 25% av opprinnelig størrelse. Dette gjør at bildet får en proporsjonal reduksjon. Istedenfor % kan vi her skrive antall pixler (målenhet på skjermen) vi vil at bildet skal få i bredde og høyde. Det er absolutt å anbefale å bruke antall pixler da man har mer kontroll over hvordan bildet vil oppføre seg ved endring av størrelsen på vinduet til nettleseren.

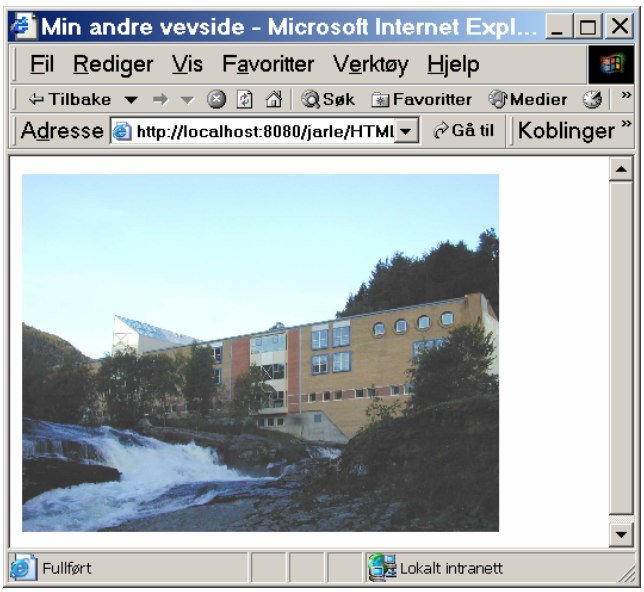
Det tredje attributtet vi skal nevne er `alt=""`. (Den legges inn i `<img>` på lik linje med `src=""`). Attributtet kommer av det engelske uttrykket "alternate text", og den skal gi besøkende som har visning av bilder slått av, eller som bruker en nettleser som ikke har mulighet for å vise bilder, en forståelse av hva bildet viser. Som eksempel kan du tenke deg et dokument som omhandler matematikk og inneholder et bilde av en graf. For besøkende som ikke kan se grafen, er det praktisk at du bruker ALT-attributtet til å fortelle dem at bildet viser en graf og oppgir funksjonen for den f.eks. f(x) = 2x + 5.

Vi har brukt bildet av Høgskulen i Sogn og Fjordane for å demonstrere bruk av bilder, og bør derfor bruke ALT-attributtet til å fortelle besøkende som ikke kan se bildet at det viser nettopp Høgskulen i Sogn og Fjordane, nærere bestemt HSF, Foss-bygget. I dokumentet vi laget i sted, kan vi nå enkelt legge dette inn i bildeelementet, og det vil da bli seende slik ut:

```
<img src="HSF.jpg" width="25%" height="25%" alt="HSF, Foss-bygget">
```

Siden vil da bli seende slik ut:

Figur 3.3: Web-side med bilde av HSF redusert til 25%



Det er av og til ønskelig at teksten i dokumentet skal flyte rundt bildet man har satt inn.

For å få lagt inn denne effekten må vi innføre et nytt attributt, `align=""`.

Ved å sette dette attributtet sammen med bilde-taggen, `<img>`-taggen, forteller vi nettleseren at bildet skal justeres til høyre eller venstre, og at teksten som følger *etter* bildet skal flyte rundt det.

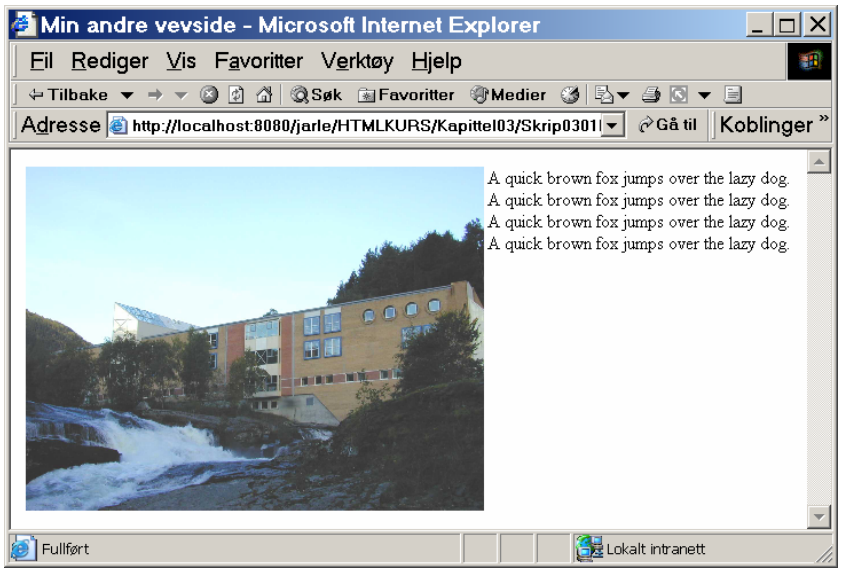
Vi tar utgangspunkt i web-siden hvor vi la inn bildet av Høgskulen i Sogn og Fjordane. For at en skal kunne se at teksten faktisk legger seg rundt bildet, må vi legge inn noe tekst i dokumentet ditt. Derfor har vi lagt inn linjen "A quick brown fox jumps over the

lazy dog" fire ganger. Du kan selvfølgelig skrive inn noe annet. Først kan du la bildet være til venstre på siden og la teksten flyte til høyre. Da legger du inn ALIGN-attributtet på denne måten, og dokumentets kropp ser slik ut:

```
<body>
  <p>
    <img src="HSF.jpg" width="400" height="300" alt="HSF, Foss-bygget" align="left">
      A quick brown fox jumps over the lazy dog.
      A quick brown fox jumps over the lazy dog.
      A quick brown fox jumps over the lazy dog.
      A quick brown fox jumps over the lazy dog.
  </p>
</body>
```

Skript0301b.html: Bilde med tekst flytende til høyre for bilde.

Nå kan du lagre endringene du har gjort og ta en titt på web-siden i nettleseren din. Den bør se omtrent slik ut:



Figur 3.4: Web-side med tekst som flyter rundt et bilde.

Her er bildet lagt til venstre og teksten flyter rundt til høyre. Hvilken side teksten ligger på i forhold til bildet, avhenger av den parameteren du velger for `align`-attributtet, `align="right"` eller `align="left"`.

Sentrering vil ikke fungere på bilder ved å sette inn "center" i ALIGN-attributtet til `<img>`. Det er fordi "center" ikke er en tillatt verdi for ALIGN-attributtet. Vi skal komme tilbake til hvordan du kan sentrere bilder i det neste kapitlet.

3.3 Tekstmanipulering og norske tegn

Til nå har teksten du har skrevet inn, blitt vist monotont og uten effekter. Dersom du er vant til å jobbe med tekstbehandling, vet du at det kan være nyttig å bruke ulike måter å framheve tekst på. Det er tre effekter som er mye brukt i tekstbehandling og som også kan brukes i HTML: <u>understreket</u> tekst, *kursivert* tekst eller **fet** tekst. I tillegg gir HTML også andre muligheter.

3.3.1 Logisk i forhold til fysisk tekstmerking

Logisk og fysisk tekstmerking kan i utgangspunktet virke komplisert, men etter hvert som du jobber med HTML, vil du se at det er enklere enn du tror. Hele tiden er det snakk om teksten i dokumentet ditt og hvordan du behandler den med HTML. *Fysisk* merking (fra engelsk markup) betyr at vi bestemmer tekstens *utseende*, det vil si understreket, *kursivert* eller **fet** skrift.

Logisk merking betyr imidlertid at vi ser bort fra utseende til teksten, og i stedet konsentrerer oss om hva vi ønsker å oppnå med merkingen. Vi har mulighet til å *framheve* teksten (på engelsk emphasise) eller **sterkt framheve** teksten (på engelsk strong emphasise).

Logisk og fysisk tekstmerking er to svært forskjellige måter å tenke på, og betydningen kommer klart fram hvis vi bruker blinde som eksempel. Blinde mennesker kan få teksten lest opp ved hjelp av talesyntese. Dersom du har merket en tekst med *kursiv* for å *framheve* noe, vil bruken av logisk tekstmerking medføre at du forteller brukeren at dette skal *framheves*, mens fysisk tekstmerking kun vil fortelle brukeren at teksten står i *kursiv*.

En grafisk nettleser (f.eks. Microsoft Internet Explorer) blir oppfordret til å vise *framhevet* tekst med *kursivert* skrift, og **sterkt framhevet** tekst med **fet** skrift. I forrige setning er teksten merket med *både* fysisk og logisk tekstmerking. Det er nettopp for å demonstrere at besøkende ikke nødvendigvis ser forskjell. Noen ganger ønsker vi at noe av teksten skal stå i kursiv, og da må vi selvsagt fortelle nettleseren det.

Meningen i det HTML-dokumentet som besøkende ser på, er imidlertid svært forskjellig om man bruker fysisk eller logisk merking. Dersom du ønsker at tekst skal framheves, oppfordres du sterkt til å benytte deg av logisk tekstmerking, ettersom du da forteller nettleseren til den besøkende at teksten faktisk skal framheves.

3.3.2 Elementer for manipulering av tekst

Det er fem vanlige elementer som du vil få bruk for ved manipulering av tekst:

Fremheving av tekst	Start-taggen er `<EM>` og slutt-taggen er `</EM>`. Taggen kommer av det engelske uttrykket "*em*phasis".
Sterk fremheving av tekst	Start-taggen er `<STRONG>` og slutt-taggen er `</STRONG>`. Taggen kommer av det engelske uttrykket "*strong* emphasis".
Kursivert tekst	Start-taggen er `<I>` og slutt-taggen er `</I>`. Taggen kommer av det engelske uttrykket "*i*talic" (på norsk "kursiv").
Fet tekst	Start-taggen er `<B>` og slutt-taggen er `</B>`. Taggen kommer av det engelske uttrykket "**bold** text" (på norsk "fet skrift").
Understreket tekst	Start-taggen er `<U>` og slutt-taggen er `</U>`. Tag'en er hentet fra det engelske uttrykket "underline" (på norsk "understreke").

Dette er de vanligste måtene å markere tekst på. Det eksisterer en god del flere elementer, men disse brukes ikke like ofte. Derfor oppfordrer vi deg til å sjekke HTML-standarden dersom du trenger andre effekter eller ønsker å se hvilke muligheter som finnes.

3.3.3 Spesielle tegn

Nå skal vi ta for oss behandlingen av de norske tegnene (æ, ø og å) samt en del andre tegn som du har bruk for. Det kan være lurt å bruke litt tid på å sette seg inn i dette avsnittet, slik at du forstår hvorfor du bør bruke disse entitetene.

Problemene vi står overfor når det gjelder bruk av de norske tegnene (æ, ø og å), henger sammen med de ulike datamaskinenes tegnsett og problemene som oppstår når dokumenter overføres mellom ulike plattformer. Et dokument som så fint ut på en PC i Norge, kan se forferdelig ut når man ser på det med f.eks. en MacIntosh i Hong Kong.

Dersom du sørger for at tegnsettet du bruker på din maskin er korrekt, er dette egentlig ikke et problem som påvirker deg direkte. Allikevel er det ikke noen stor jobb å søke og erstatte de norske tegnene med koder i dokumentene dine, og det gjør det enklere for besøkende å få med seg informasjonen på web-sidene dine. I stedet for æ, ø og å bruker man følgende koder:

Norsk tegn	**Tekstentitet**	**Numerisk entitet**
æ	&aelig;	&#230;
ø	&oslash;	&#248;
å	&aring;	&#229;
Æ	&AElig;	&#198;
Ø	&Oslash;	&#216;
Å	&Aring;	&#197;

Tabell 3.1: Numeriske entiteter for nordiske tegn

Disse kodene kalles "entiteter", og alle starter med tegnet "&" og avsluttes med et semikolon.

Entiteter kommer i to varianter, en basert på tekst og en basert på en numerisk verdi. Den første varianten er lettere å forstå og huske, men det er bare den siste som garantert fungerer slik man ønsker. Det er fordi en del nettlesere ikke støtter tekstentitetene, mens alle nettlesere støtter den numeriske entiteten. Legg merke til at tekstentitetene skiller mellom hvorvidt du begynner entiteten med liten eller stor bokstav. Dette er et unntak fra regelen om at HTML ikke skiller mellom store og små bokstaver.

I tillegg til disse er det fire andre entiteter som du får bruk for:

Tegn	Tekstentitet	Numerisk entitet
<	&lt;	&#60;
>	&gt;	&#62;
&	&amp;	&#38;
" (anførselstegn)	&quot;	&#34;

Tabell 3.2: Numeriske entiteter for spesialtegn.

"<" brukes til å starte tagger og, ">" brukes til å avslutte dem. Det har du allerede sett og brukt mange ganger. Dersom du hadde skrevet dem alene inn i dokumentet, ville en nettleser kunne tro at du ønsket å starte eller avslutte en tagg, og dermed skape problemer. Hvis du f.eks. ønsker å skrive eksempler på HTML i et dokument, slik det er gjort i dette kurset, er du nødt til å bruke entiteter, ellers vil nettleseren tolke eksemplene som HTML og dermed gjøre dem usynlige.

Grunnen til at man må bruke en entitet for "&", er at den brukes til å starte entiteter og derfor kan skape problemer om man lar den stå alene. Når det gjelder anførselstegn, må du bruke entiteten for å unngå forveksling mellom anførselstegn i attributter og anførselstegn i teksten.

3.4 Lister

HTML gir deg mulighet til å sortere informasjon i tre typer lister:

1. Nummerert liste
2. Unummerert liste (punktliste)
3. Definisjonsliste

En nummerert liste ser ut som listen ovenfor. Elementet er et blokknivåelement der start-taggen er `<ol>` og slutt-taggen er `</ol>` (fra det engelske uttrykket "ordered list"). Dersom du bruker Netscape Navigator versjon 4.0 eller nyere, lurer du kanskje på hvorfor et blokknivåelement (som listen ovenfor) ikke vises med en tom linje over og under? Programmer har en tendens til å inneholde feil, og det er også tilfelle med Navigator. Dette skyldes at implementasjonen av stilsett i Navigator er noe mangelfull, og dette er et av problemene. Dersom du bruker Internet Explorer 4.0 eller nyere, vil elementet vises slik det skal ettersom implementasjonen av stilsett her er bedre. Igjen en påminnelse om at en web-side ikke er garantert å se ut slik du hadde tenkt.

Unummererte lister kan grupperes sammen med nummererte lister fordi likheten mellom dem er stor. En unummerert liste er også et blokknivåelement, der start-taggen er `<ul>` og slutt-tag'en er `</ul>` (fra det engelske uttrykket "unordered list").

Det som er felles for nummererte og unummererte lister, er måten man legger inn informasjon i dem. Mellom start- og slutt-taggene kan det bare legges inn listepunkter

(fra det engelsk uttrykket "list items"). Listepunktene kan imidlertid inneholde avsnitt med tekst, bilder, tabeller, en ny liste osv. Et listepunkt har en egen tagg, `<li>`, som ikke har noen slutt-tagg. For å forstå forskjellen mellom de ulike listene og hva de brukes til, skal vi vise eksempler på en nummerert liste, en unummerert liste og en definisjonsliste og gi HTML-kodene for disse.

3.4.1 Nummerert liste - og

Ta fram web-siden du arbeidet med under avsnittet om bilder, "andreside.html", og fjern avsnittet med bildet slik at dokumentets kropp igjen blir tom. I det tomme dokumentet skal du sette inn en nummerert liste. La oss lage et eksempel med oppgaver i matematikk fra barneskolen. Listen startes med `<ol>` og avsluttes med `</ol>`. Mellom disse skriver du inn listepunktene, som startes med `<li>`. Som eksempel kan du skrive inn følgende liste:

```
<ol>
 <li>2 + 5 = ?</li>
 <li>9 - 2 = ?</li>
 <li>6 + 3 = ?</li>
 <li>4 + 4 = ?</li>
</ol>
```

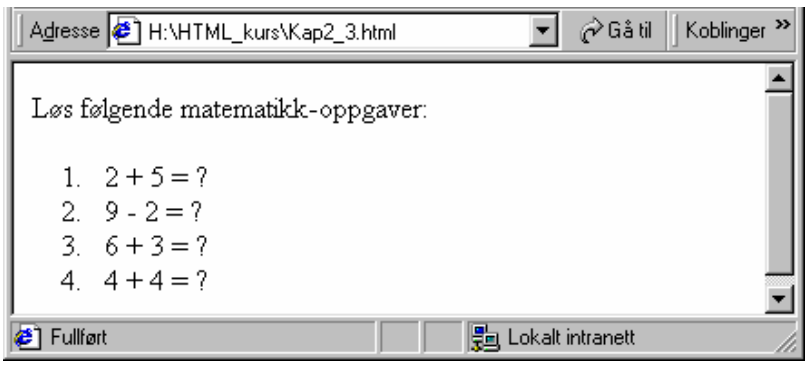
Vi legger til litt tekst for å gjøre eksemplet mer realistisk, og dokumentets kropp vil da se omtrent slik ut:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
   <title>Min tredje vevside</title>
  </head>

  <body>
  <p>L&oslash;s f&oslash;lgende matematikk-oppgaver:</p>
  <ol>
      <li>2 + 5 = ?</li>
      <li>9 - 2 = ?</li>
      <li>6 + 3 = ?</li>
      <li>4 + 4 = ?</li>
  </ol>
  </body>
</html>
```

Skript0302a.html: Bilde med tekst flytende til høyre for bilde.

Skriv inn koden og arkiver dokumentet som skript0302.html. Legg her merke til at det i skriptet over er benyttet tekstentiteten for bokstaven ø. Ta så en titt på hvordan vevsiden tar seg ut i nettleseren din. Resultatet skal se omtrent slik ut:

Figur 3.5: Web-siden med en nummerert liste.



En nummerert liste behøver ikke nødvendigvis sorteres med arabiske tall.

Den kan også sorteres med små eller store bokstaver (a, b, c osv. eller A, B, C osv.) og små eller store romertall (i, ii, iii osv. eller I, II, III osv). For å få til dette endres start-taggen til elementet i tråd med tabellen under:

	Tilhørende visning	**Start-tagg**
Arabiske tall	1, 2, 3 …	<ol type="1">
Små bokstaver	a, b, c …	<ol type ="a">
Store bokstaver	A, B, C …	<ol type ="A">
Små romertall	i, ii, iii …	<ol type ="i">
Store romertall	I, II, III …	<ol type ="I">

Tabell 3.3:Attributter for visning av nummererte lister

3.4.2 Unummerert liste (punktliste)

En unummerert liste har mye til felles med en nummerert liste. Forskjellen ligger i at en unummerert liste ikke har mulighet for arabiske tall, romertall og bokstaver, men i stedet ordner listen med punkter. Start-taggen for en unummerert liste er `<ul>` og slutt-taggen er `</ul>`. Igjen må du huske at en liste av denne typen kun kan inneholde listepunkter, og at disse igjen skal inneholde informasjonen.

Nå kan du legge inn et eksempel på en unummerert liste på web-siden din. Listen kan f.eks. inneholde stikkord fra en tekst. Igjen tar du fram web-siden din, sletter innholdet i dokumentets kropp og legger inn den nye listen. Koden til listen kan se slik ut:

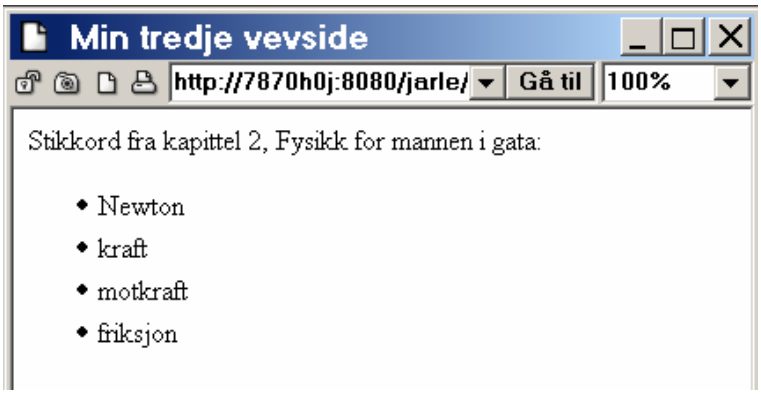
```
<ul>
  <li>Newton</li>
  <li>kraft</li>
  <li>motkraft</li>
  <li>friksjon</li>
</ul>
```

Når du har satt dette inn i dokumentets kropp, vil den se omtrent slik ut (igjen har vi brukt forklarende tekst til listen):

```
<body>
  <p>Stikkord fra kapittel 2, Fysikk for mannen i gata:</p>
    <ul>
          <li>Newton</li>
          <li>kraft</li>
          <li>motkraft</li>
          <li>friksjon</li>
    </ul>
</body>
```

Skript0302b.html: Bilde med tekst flytende til høyre for bilde.

Nå kan du lagre endringene du har gjort og deretter se på web-siden i nettleseren din. Da kan det se slik ut:



Figur 3.6: Web-siden med en unummerert liste.

I likhet med nummerte lister kan du også her benytte et tilleggsattributt for å angi hvilket symbol du vil benytte som "bullet".. For å få til dette kan du endre start-taggen til elementet i tråd med tabellen under:

"Bullet"	Start-tagg
• standard bullett	
○ sirkel-bullet	<ul type="circle">
▪ firkant bullet	< type="**square**">

Tabell 3.4:Attributter for visning av unummererte lister

Det er her verdt å merke seg at hvordan punktene vises kan variere noe fra nettleser til nettleser.

3.4.3 Definisjonsliste

En definisjonsliste er en type liste som du kan bruke når du vil definere ord og uttrykk. Igjen er listen et blokknivåelement, og som for de to foregående listene er det en del begrensninger å forholde seg til. Du husker kanskje at de andre listene kun kan inneholde listepunkter? Definisjonslister har en lignende begrensning som det er viktig at du er klar over.

Start-taggen for en definisjonsliste er `<dl>`, og slutt-taggen er `</dl>`, hentet fra det engelske uttrykket "definition list". Innholdet i en definisjonsliste er ganske spesielt. Listen kan kun inneholde to elementer, `<dt>` og `<dd>`, og disse skal passe sammen i par. `<dt>` står for "definition term" og skal inneholde en term (ord eller uttrykk) som skal forklares. `<dd>` står for "definiton description" og er den korre-sponderende definisjonen til termen som `<dt>` inneholder.

I motsetning til ved nummererte og unummererte lister er det også begrensninger i hva disse to kan inneholde. Termen kan kun være tekst, mens definisjonen kan inneholde det du måtte ønske (avsnitt, tabeller, bilder osv).

Det er lettere å illustrere dette med et eksempel. Vi ønsker å besvare spørsmålet "hva er HTML?" Koden ser slik ut:

```
</body>
  <dl>
    <dt>Hva er HTML?</dt>
        <dd>HTML er et kunstig språk beregnet på strukturering av
            informasjon i sk. 'hypertext'-dokumenter. Stikk i strid
            med en vanlig oppfatning er ikke HTML en måte å beskrive
            utseendet på hjemmesider.
        </dd>
  </dl>
</body>
```

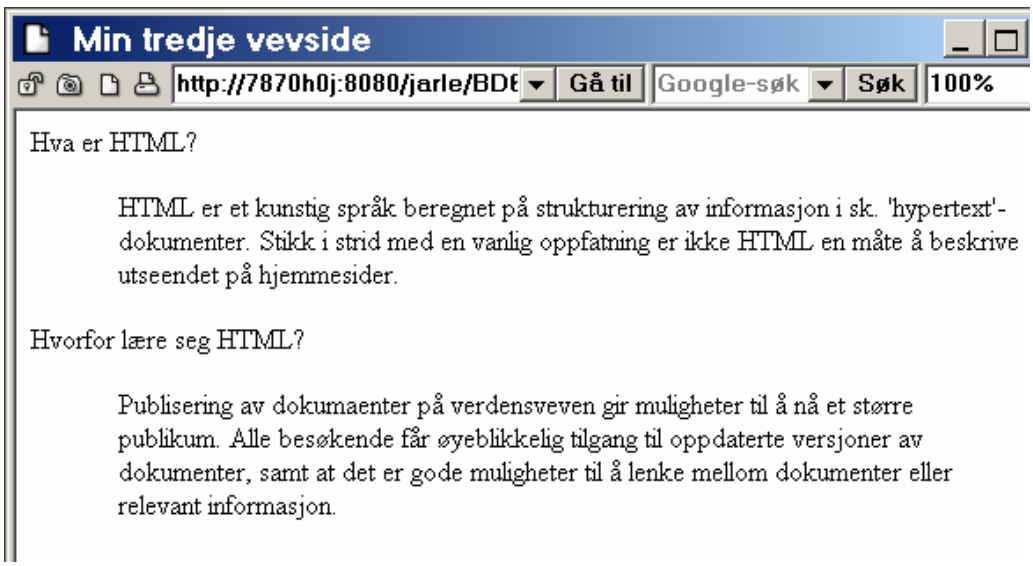
Skript0302c.html: Bilde med tekst flytende til høyre for bilde

Finn fram dokumentet " Skript0302b.html " og fjern den unummererte listen, slik at dokumentets kropp igjen blir tomt. Deretter skriver du inn eksemplet ovenfor. Når du har gjort det, lagrer du endringene, som Skript0302c.html, og tar en titt på det med nettleseren . Resultatet vil se omtrent slik ut:

Figur 3.7: Web-side med eksempel på definisjonsliste

På figuren ser vi én måte å vise fram en definisjonsliste på. Ordet/uttrykket vi definerer står på én linje, mens definisjonen er rykket inn og følger direkte under. Dersom vi hadde definert flere ord/uttrykk ville ordet/uttrykket stått på en ny linje under forrige ord/uttrykk, og en ny definisjon på nytt under der igjen. Et eksempel er vist i figur 3.7. Framvisningsmåten kan imidlertid variere avhengig av hva slags plattform den besøkende har.

Figur 3.7: Web-side med eksempel på en definisjonsliste med flere punkter, vist i nettleseren Opera. (skript0302d.html)



3.4.4 Nøsting av lister

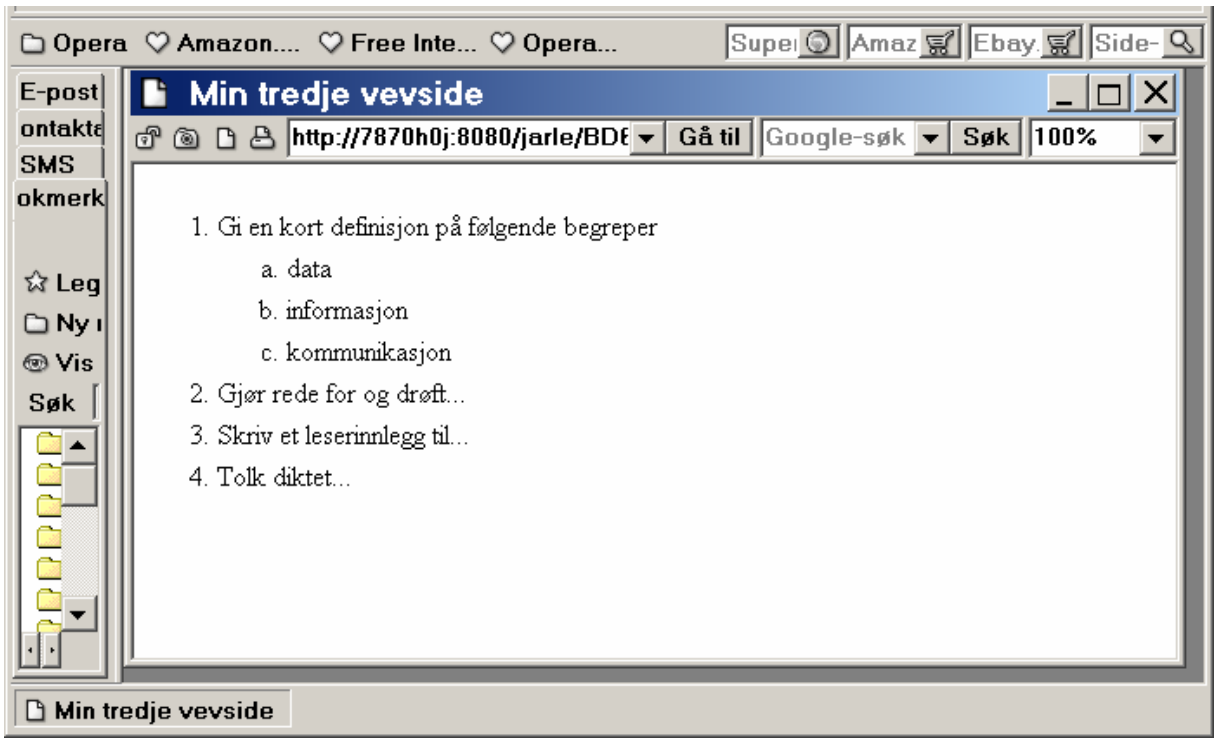
Det er selvfølgelig mulig å nøste lister i hverandre, det vil si å bruke en liste som et punkt i en annen liste. Det kan du bruke f.eks. når du har en nummerert liste med oppgaver (1, 2, 3 ...) og ønsker at oppgave 1 skal ha tre delspørsmål: a, b og c. Som kjent kan en nummerert eller unummerert liste kun inneholde listepunkter, men det enkelte listepunktet kan igjen inneholde en ny liste. I vårt eksempel vil vi ha en nummerert liste hvor første listepunkt skal inneholde en ny nummerert liste med punktene a, b og c.

Fjern den forrige listen i dokumentet ditt (definisjonslisten), sett inn koden under i stedet for, og arkiver endringene som skript0302e.html.

```
<body>
  <ol>
     <li> Gi en kort definisjon på følgende begreper</li>
     <ol type="a">
       <li>data</li>
       <li>informasjon</li>
       <li>kommunikasjon</li>
     </ol>
     <li>Gjør rede for og drøft...</li>
     <li>Skriv et leserinnlegg til...</li>
     <li>Tolk diktet...</li>
  </ol>
</body>
```

Skript0302e.html: Bilde med tekst flytende til høyre for bilde

Når du så lagrer endringene og ser på dokumentet med nettleseren , kan det se slik ut:



Figur 3.8: Web-side med eksempel på nøstede lister

3.5 Sitater

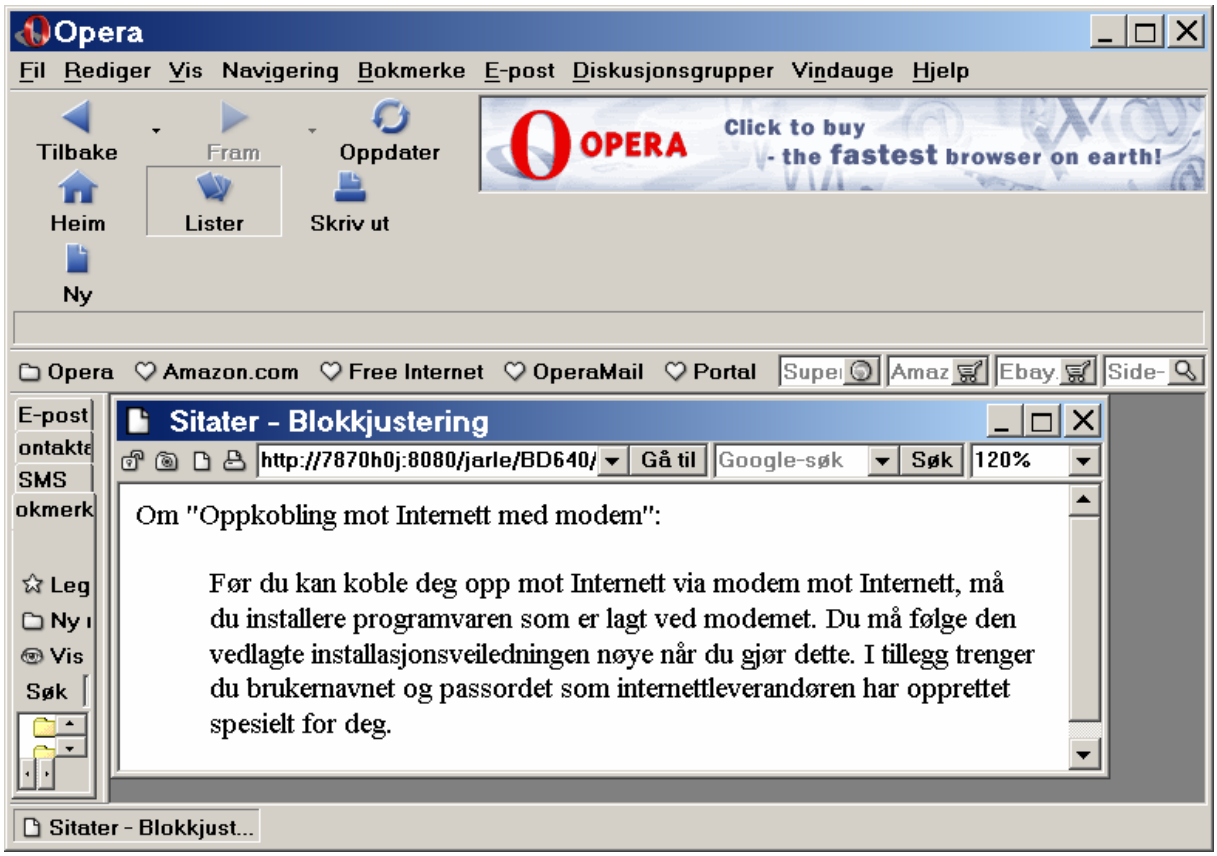
Det finnes et eget element som brukes til å definere sitater. Start-taggen er `<blockquote>`, slutt-taggen er `</blockquote>` og elementet er et blokknivå-element.

Måten sitatet vises fram på varierer, men det er nokså vanlig at de ulike nettleserne øker margene på høyre og venstre side. Dette er enkelt å demonstrere med et eksempel. Du sletter igjen listene fra forrige eksempel og skriver inn koden i dokumentets kropp, arkiver det endrede dokumentet som skript0303.html:

```
<body>
   <p>Om &quot;Oppkobling mot Internett med modem&quot;:</p>
   <blockquote>Før du kan koble deg opp mot Internett via modem mot Internett,
       må du installere programvaren som er lagt ved modemet.
       Du må følge den vedlagte installasjonsveiledningen nøye når du gjør
       dette.I tillegg trenger du brukernavnet og passordet som
       internettleverandøren har opprettet spesielt for deg.
   </blockquote>
</body>
```

Skript0303.html: Eksempel på sitat

Når du har skrevet inn sitatet i dokumentet ditt, lagrer du endringene og tar en titt på det med nettleseren . Resultatet kan se slik ut:



Figur 3.9: Web-side med eksempel på sitat, slik den vises i Opera.

3.6 Tabeller

I HTML finnes det også en mulighet for å lage tabeller slik som i en vanlig tekstbehandler. Koden for tabeller er ganske enkel, men blir fort uoversiktlig. Tabeller har i utgangspunktet fire bestanddeler med ulike egenskaper. Når det gjelder selve tabellen, er den et blokknivåelement. La oss se på en tabell med de forskjellige byggeklossene den har:

Element	Start-tagg	Slutt-tagg
Tabell	<TABLE>	</TABLE>
Tabellrekke	<TR>	</TR>
Celle m/overskrift	<TH>	</TH>
Tabellcelle	<TD>	</TD>

Tabell 3.4:Grunnleggende byggeklosser i en tabell

For å demonstrere hvordan en tabell bygges opp, kan vi lage en timeplan. Den skal inneholde alle hverdagene (mandag til fredag), og hver dag skal bestå av 6 skoletimer.

Når du skal lage tabellen, må du tenke i rader og kolonner. Vær oppmerksom på at i HTML er tabellen inndelt i rader som så er inndelt i celler. Cellene kan være enten vanlige celler eller overskriftsceller. Overskriftscellene er ment å stå på toppen av en kolonne eller i begynnelsen av en rad.

La oss begynne med å lage selve tabellen. Start-taggen kommer i utgangspunktet i to versjoner, en for tabeller uten rammer og en for tabeller med rammer. Uten rammer er start-taggen `<TABLE>`, og med rammer er start-taggen `<TABLE BORDER>`. Slutt-taggen er imidlertid lik for begge versjonene, nemlig `</TABLE>`. Til timeplanen vår vil vi ha en tabell med rammer, og koden for selve tabellen blir da slik:

```
<table border>

</table>
```

Det vi har gjort nå, er å sette opp det ytterste rammeverket for tabellen, og for å fylle den må vi bruke andre tagg-elementer. For å bygge opp en tabell bør du starte med øverste rad og jobbe deg nedover. I vår tabell ønsker vi en topprad med overskrifter med teksten "Time" ytterst til venstre, og deretter ukedagene bortover mot høyre, slik som dette:

Figur 3.10: Overskriftsraden i en timeplan.

For å få til det må du bruke denne koden:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Tabeller - timeplaneksempel</title>
</head>
<body>
<table border>
    <tr>
      <th>Time</th><th>Mandag</th><th>Tirsdag</th><th>Onsdag</th>
      <th>Torsdag</th><th>Fredag</th>
    </tr>
    </table>
</body>
</html>
```

Skript0304.html: Koden for overskriftsrad i timeplan.

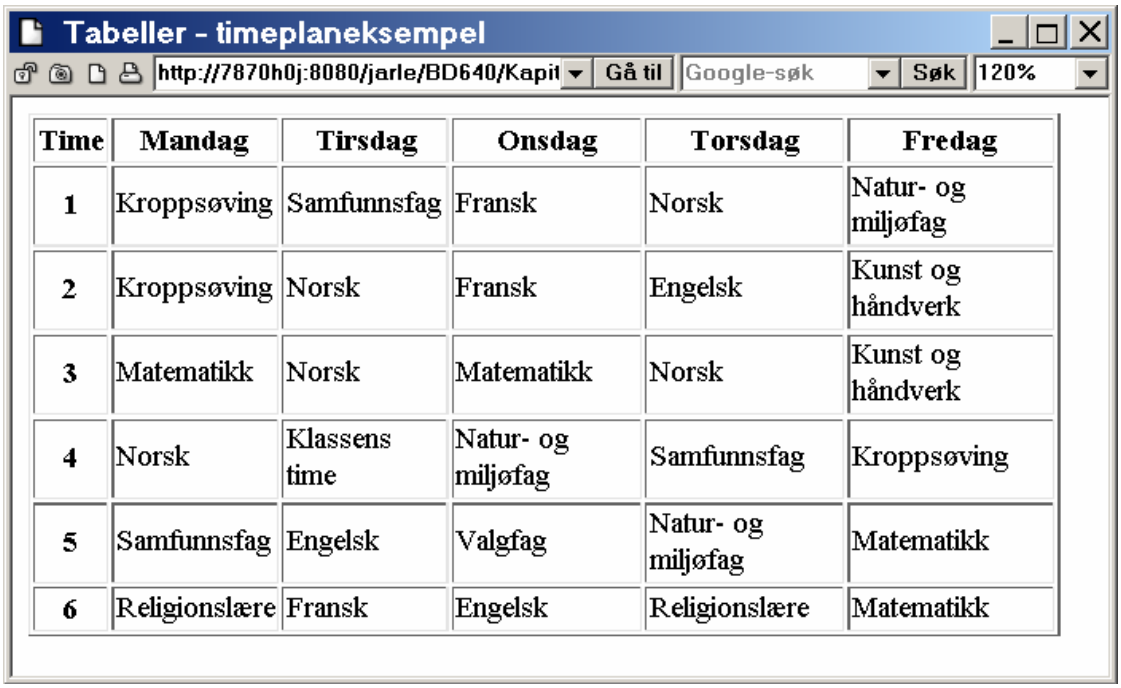
Dette er koden for en tabell (innenfor `<table border>` og `</table>`) som inneholder en rad (markert med `<tr>` og `</tr>`). Tabellraden inneholder en del celler, der hver celle er en overskriftscelle og derfor er markert med `<th>` og `</th>`. Vi får da et resultat som vist i avsnittet over koden. Skriv nå inn koden over og arkiver denne som skript0304.html.

Vi ønsker nå å utvide tabellen vår slik at den lister opp timene 1 til 6 på venstre side, under "Time". Deretter vil vi legge inn fagene under hver dag, slik at det passer med hvilken time faget undervises i. Koden ser da slik ut:

```
<table border>
  <tr>
     <th>Time</th><th>Mandag</th><th>Tirsdag</th><th>Onsdag</th><th>Torsdag</th><th>Fredag</th>
  </tr>
  <tr>
     <th>1</th>
     <td>Kroppsøving</td><td>Samfunnsfag</td><td>Fransk</td><td>Norsk</td><td>Natur- og miljøfag</td>
  </tr>
  <tr>
     <th>2</th>
     <td>Kroppsøving</td><td>Norsk</td><td>Fransk</td><td>Engelsk</td><td>Kunst og håndverk</td>
  </tr>
  <tr>
     <th>3</th>
     <td>Matematikk</td><td>Norsk</td><td>Matematikk</td><td>Norsk</td><td>Kunst og håndverk</td>
  </tr>
  <tr>
     <th>4</th>
     <td>Norsk</td><td>Klassens time</td><td>Natur- og miljøfag</td><td>Samfunnsfag</td><td>Kroppsøving</td>
  </tr>
  <tr>
     <th>5</th>
     <td>Samfunnsfag</td><td>Engelsk</td><td>Valgfag</td><td>Natur- og miljøfag</td><td>Matematikk</td>
  </tr>
  <tr>
     <th>6</th>
     <td>Religionslære</td><td>Fransk</td><td>Engelsk</td><td>Religionslære</td><td>Matematikk</td>
  </tr>
</table>
```

Skript0304b.html: Koden for timeplan.

Legg til fagene i skript0304.html, og arkiver det hele som skript0304bg.html. Etter at du har lagret dokumentet, kan du se på det med nettleseren din. Tabellen vil se omtrent slik ut:



Time	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
1	Kroppsøving	Samfunnsfag	Fransk	Norsk	Natur- og miljøfag
2	Kroppsøving	Norsk	Fransk	Engelsk	Kunst og håndverk
3	Matematikk	Norsk	Matematikk	Norsk	Kunst og håndverk
4	Norsk	Klassens time	Natur- og miljøfag	Samfunnsfag	Kroppsøving
5	Samfunnsfag	Engelsk	Valgfag	Natur- og miljøfag	Matematikk
6	Religionslære	Fransk	Engelsk	Religionslære	Matematikk

Figur 3.11: Web-side med eksempel timeplan som en tabell.

Tabellen lages ovenfra og nedover. Den eneste store forskjellen du skal legge merke til mellom den første tabellen, som bare inneholdt overskriftsceller, og den nye, er introduksjonen av tabellceller (markert med `<td>` og `</td>`). I tillegg er tabellen blitt ganske stor og dermed mindre oversiktlig. Mange HTML-verktøy har egne tabell-editorer som lar deg redigere tabeller som et regneark. Det gjør det mer oversiktlig å lage tabeller.

4. Litt mer avanserte emner i HTML

4.1 Mål

Etter å ha gjennomgått dette kapittelet, skal du:

- kjenne til elementene for subskript og superskript - <SUB> og <SUP>
- kunne sentrere tekst, bilder og lignende
- kunne lage lenker innad i dokumenter
- kunne bruke bilder som lenke
- vite forskjell på relativ og absolutt adressering
- vite hvordan du bruker <META> til å beskrive et dokument

4.2 Subskript og superskript

Elementene for subskript og superskript kunne vært nevnt under avsnittet om tekstmanipulering i studieenhet 2, men vi har valgt å gjennomgå dem her fordi behovet for dem ikke oppstår så ofte som de andre tekstelementene. De to elementene fungerer akkurat som i en tekstbehandler. Subskript gir deg muligheten til å skrive inn tekst som havner 1/2 linje under vanlig tekst, slik som i kjemiske symboler som H_2O.

Superskript gir deg muligheten til å skrive inn tekst som havner 1/2 linje over vanlig tekst, slik at du kan få skrevet f.eks. TM (TradeMark) slik det skal se ut: Bottolfburger™. Et annet nærliggende eksempel er bruk av potenser i matematikken. For å få skrevet likningen $f(x) = x^4 - 2x^5$ benytter du superskript.

Subskript har start-taggen `<sub>` og slutt-taggen `</sub>`, og superskript har start-taggen `<sup>` og slutt-taggen `</sup>`. Begge elementene er tekstnivåelementer.

4.3 Preformatert tekst

Forklaringen på preformatert tekst ligger i navnet: det er tekst som allerede er formatert. I HTML betyr det at nettleserne skal ta hensyn til bruk av mellomrom og linjeskift i det avsnittet som er markert som preformatert tekst. Start-taggen for preformatert tekst er `<pre>` og slutt-taggen er `</pre>`. Preformatert tekst er et blokknivåelement.

Det kan være nyttig å bruke elementet for preformatert tekst når du ønsker å vise programmeringskode eller andre områder hvor innrykk er viktig. Vi skal se på et eksempel, men i stedet for programmeringskode skal vi se på HTML. I kapittel 2 så vi hvordan vi fikk laget lister, og vi viste et eksempel på en nummerert liste. Eksemplet så slik ut:

```
<ol>
  <li>2 + 5 = ?</li>
  <li>9 - 2 = ?</li>
  <li>6 + 3 = ?</li>
  <li>4 + 4 = ?</li>
</ol>
```

Som vi ser er taggene for listepunktene rykket inn to tegn. For å få til det har vi brukt elementet for preformatert tekst, og koden ser slik ut:

```
<body>
<pre>&lt;ol&gt;
  &lt;li&gt;2 + 5 = ?
  &lt;li&gt;9 - 2 = ?
  &lt;li&gt;6 + 3 = ?
  &lt;li&gt;4 + 4 = ?
&lt;/ol&gt;</pre>
</body>
```

Skript0401.html: Eksempel på pre-formattert tekst

Det er altså koden for preformatert tekst som sørger for at teksten rykkes inn to tegn. På lignende måte kan du også benytte deg av preformatert tekst i andre sammenhenger. Det er imidlertid viktig at du ikke misbruker muligheten til for eksempel å få innrykk i teksten du skriver. De ulike nettleserne vil ikke brekke om linjer (dvs. endre layout) med preformatert tekst. Dermed kan web-siden din være vanskelig å lese for besøkende hvis du for eksempel har brukt lange linjer. Det er fordi linjene da kan falle utenfor en besøkendes skjermbilde. Preformatert tekst skal derfor brukes varsomt.

4.4 Sentrering

Så langt har alt vi har gjort vært justert til venstre (rett venstremarg), bortsett fra bilder, som vi også har vist at kan justeres til høyre (rett høyremarg) ved hjelp av `<img align="right">`. Iblant er det imidlertid behov for å sentrere eller høyrejustere tekst, overskrifter eller lignende.

De fleste elementene du har behov for å sentrere, aksepterer at du bruker et eget attributt til å gjøre dette. Dette attributtet er `align=""`, og den tar tre forskjellige parametre: "left", "center" og "right". Effekten av dem er at teksten vises henholdsvis med rett venstremarg, sentrert og med rett høyremarg. La oss sentrere en overskrift, et avsnitt og en tabell som eksempler på hvordan dette virker. Her er koden vi bruker:

```
<body>
  <h1 align="center">Sentrert overskrift</h1>

  <p align="center">Sentrert avsnitt.</p>

  <table border align="center">
   <tr><td>Sentrert tabell.</td></tr>
  </table>
<body>
```

Skript0402a.html: Sentrering ved hjelp av align-attributtet

Du bør sette inn disse kodene i en ny fil som du lagrer som skript0402.html. Husk at også denne filen må inneholde deklarasjon, dokumenthode og dokumentkropp, og kodene over skal settes inn i dokumentets kropp. Når du har lagret filen, kan du se på den i nettleseren din. Resultatet kan se slik ut:

Figur 4.1: Web-side med sentrerte elementer, slik den vises i nettleseren.

Du kan ikke sentrere bilder ved å sette inn "center" i ALIGN-attributtet til `<img>`, fordi bilder ikke er egne avsnitt. Dessuten er ikke "center" en tillatt verdi for ALIGN-attributtet i kombinasjon med `<img>`. For å sentrere et bilde må du derfor lage et avsnitt som inneholder bildet og deretter sentrere det. Da vil innholdet i avsnittet, som altså er bildet, bli sentrert.

La oss lage et avsnitt hvor vi sentrerer et bilde. Vi benytter bildet av Høgskulen i Sogn og Fjordane, som vi brukte i kapittel 3 (HSF.jpg). Koden blir slik:

```
<body>
   <h1 align="center"> Sentrering av bilde </h1>
   <p align="center">
      <img src="HSF.jpg" width="400" height="300" alt="HSF, Foss-bygget">
   </p>
</body>
```

Skript0402b.html: Sentrering ved hjelp av align-attributtet

Skriv koden inn og akrkiver skriptet som" Skript0402b.html", vil resultatet kunne se slik ut (den sentrerte overskriften er kun med for å vise klarere at bildet faktisk er sentrert):

Figur 4.2: Web-side med eksempel på sentrert bilde

4.5 Mer om lenker

I dette avsnittet skal vi gjennomgå hvordan du kan lage lenker direkte til avsnitt i dokumenter, og hvordan du kan bruke et bilde som lenke. I tillegg skal vi forklare forskjellen mellom absolutt og relativ adressering.

4.5.1 Lenker direkte til avsnitt i dokumenter

For å kunne lenke direkte til et avsnitt i et dokument må du først markere hvor i dokumentet lenken skal treffe. Til dette bruker man ankerelementet (<a>) og name-attributtet. La oss lage et avsnitt som vi kan lenke til. For å ha noe mer tekst å jobbe med kan vi ta utgangspunkt i skript0302d.html. Koden for avsnittet blir slik:

```
<head>
  <title>Eksempel på interne lenker</title>
</head>
<body>
    <dl>
      <dt>Hva er HTML?</dt>
      <dd><p>HTML er et kunstig språk beregnet på strukturering av
             informasjon i sk. 'hypertext'-dokumenter. Stikk i strid
             med en vanlig oppfatning er ikke HTML en måte å beskrive
             utseendet på hjemmesider.</p>
      </dd>
      <dt>Hvorfor lære seg HTML?</dt>
      <dd><p>Publisering av dokumaenter på verdensveven gir muligheter
             til å nå et større publikum. Alle besøkende får øyeblikkelig
             tilgang til oppdaterte versjoner av dokumenter, samt at det er
             gode muligheter til å lenke mellom dokumenter eller relevant
             informasjon.</p>
      </dd>
    </dl>
    <p><a name="lenk_meg">Dette avsnittet </a>
      kan du lage en lenke til dersom du ønsker.
      Fint, ikke sant?
    </p>
</body>
```

Skript0403.html: Lage et "bokmerke" i HTML-tekst

I kapittel 2 brukte vi <a href=""> til å lage en lenke til et annet dokument. Ved å bruke name- attributte t <a name=""> kan en lage lenker som går til spesifikke avsnitt i dokumentet. Når dokumentene man jobber med begynner å bli store, kan det være praktisk å lenke fra for eksempel innholdsfortegnelsen og rett til avsnittene som er nevnt i den. Det er nettopp dette <a name=""> gir oss muligheten til.

La oss anta at vi har en del vevsider liggende på URL'en "http://stud-vevtjener/~brukernavn/". Vi skal skrive en stor prosjektoppgave, der et av kapitlene i prosjektdokumentet ligger i filen "kapittel_4.html". I dette kapittelet har vi laget et avsnitt i teksten hvor taggen <a name="avsnitt_3"> er brukt. Denne taggen gir oss nå altså mulighet til å lenke direkte til dette avsnittet.

For å kunne lenke direkte til avsnittet må vi kjenne URL'en til avsnittet. Fra før vet vi at URL'en til web-sidene er "http://stud-vevtjener/~brukernavn/" og at avsnittet vi ønsker å lenke til, ligger i filen "kapittel_4.html". URL'en til filen blir da

"http://stud-vevtjener/~brukernavn/kapittel_4.html". Vi forutsetter da at vi har lagret filen "kapittel_4.html" i samme katalog/mappe som de andre web-sidene våre. For å lage en lenke til avsnittet bruker vi URL'en til selve dokumentet, legger til et skigardtegn, #, og til slutt den verdien som står i anførselstegn i `<a name="">`. I vårt tilfelle er denne verdien "avsnitt_3". Dermed kjenner vi URL'en til det avsnittet vi ønsker å lenke til, og koden for lenken ser da slik ut:

```
<a href="http://stud-vevtjener/~brukernavn/kapittel_4.html#avsnitt_3">
Lenke direkte til avsnitt 3</a>
```

På lignende måte kan vi lage en lenke til avsnittet med `<a name="lenk_meg">` som vi laget i begynnelsen av dette kapitlet. Vi antar at du har lagret filen med avsnittet som "skript0403.html", og da blir koden for lenken slik:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Eksempel på lenk til internt bokmerke</title>
</head>
<body>
      <a href=" skript0403.html#lenk_meg">
      Lenke direkte til eksempelavsnittet</a>
</body>
</html>
```

Skript0403b.html: Lenke til et "bokmerke" i HTML-tekst

For å test ut dette kan du lage et lite testdokument hvor du legger inne koden over. Arkiver skriptet som skript0403b, og test ut om det funker.

4.5.2 Bilde som lenke

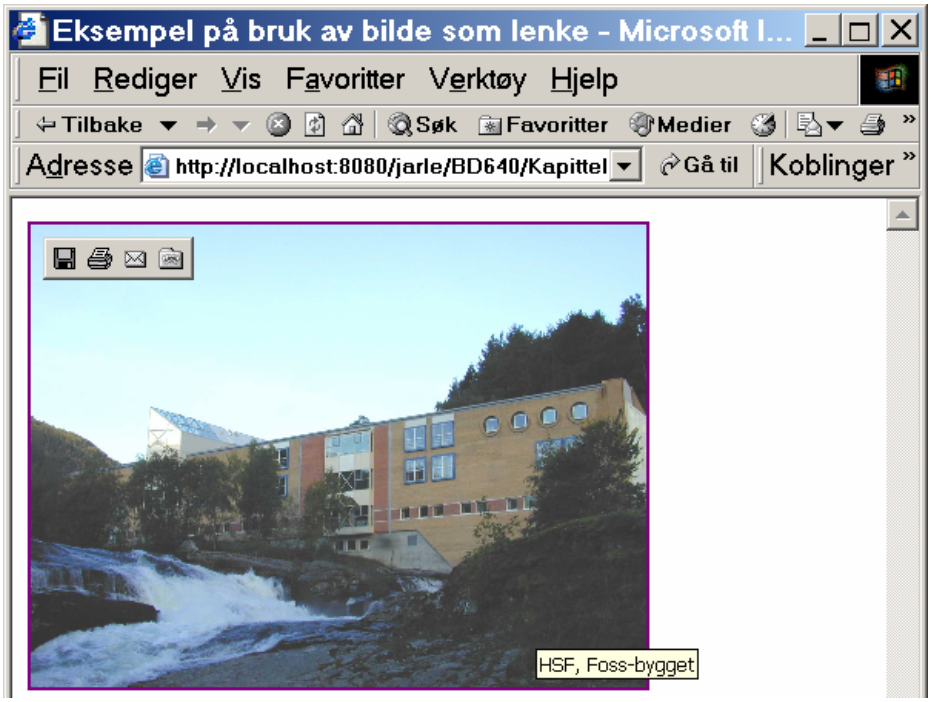
Noen ganger kan det være nyttig å lage en lenke ved å bruke et bilde i stedet for en forklarende tekst. Dermed vil besøkende, dersom de har muligheten til det, kunne klikke på bildet og få opp dokumentet lenken peker til. En slik lenke kan du lage på vanlig måte ved å bruke `<a href="">`. I stedet for å legge inn tekst mellom start- og slutt-taggene skriver du inn taggen for et bilde.

Vi kan bruke et bilde du allerede har lagret, nemlig bildet av Høgskulen i Sogn og Fjordane, og lage en lenke til Høgskulen i Sogn og Fjordane ved hjelp av det. Koden for det blir slik:

```
<body>
  <a href="http://www.hisf.no">
  <img src="hsf.jpg" width="400" height="300" alt="HSF, Foss-bygget"></a>
</body>
```

Skript0403c.html: Bruk av bilde som lenke

Nå kan du bytte ut innholdet i dokumentets kropp i filen "skript0403b.html" og legge inn koden over. Arkiver så dette dokumentet som skript0403c.html. Da kan det se slik ut:



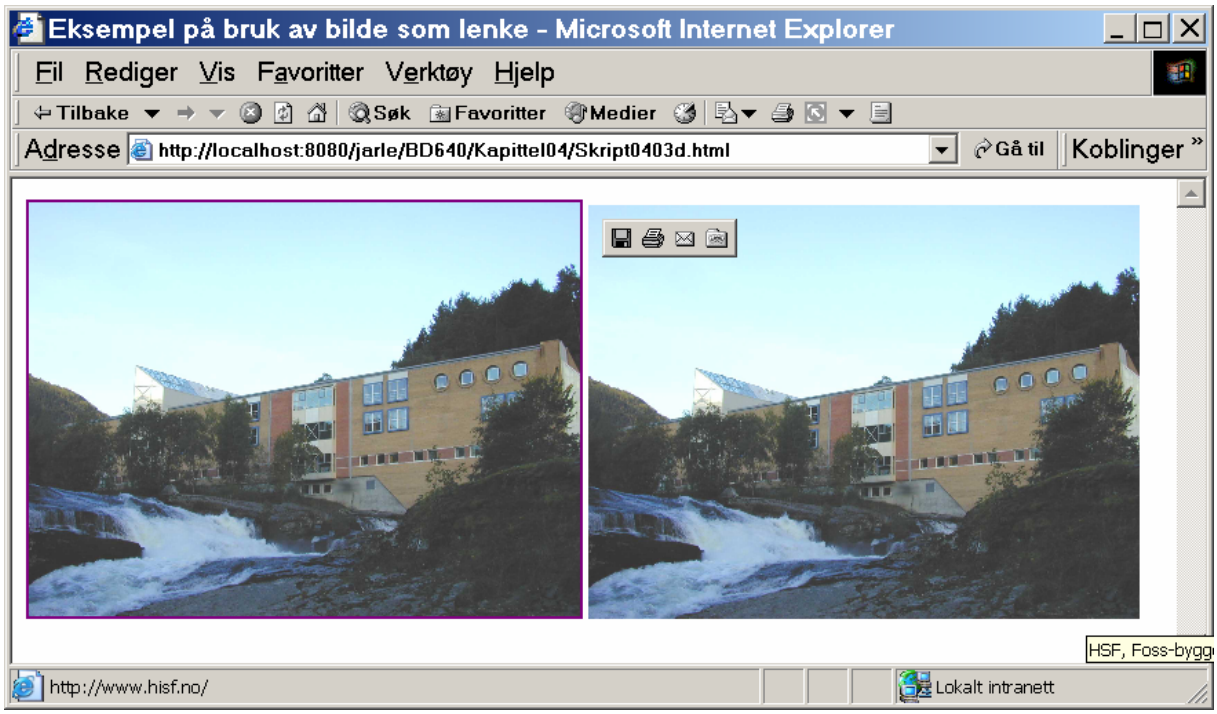
Figur 4.3: Web-side med et bilde som lenke.

Rammen rundt bildet viser den besøkende at bildet er en lenke og at man kan klikke på det for å komme dit lenken peker. Det er også mulig å lage lenker med bilder uten ramme. Det gjør du ved å legge til `border`-attributtet i `<img>`, slik:

```
<a href="http://www.hisf.no">
<img src="hsf.jpg" width="400" height="300" alt="HSF, Foss-bygget"
border="0"></a>
```

Skript0403d.html: Bruk av bilde uten ramme som lenke

Når du skriver inn denne koden i filen din og ser på den med nettleseren, kan det se slik ut:



Figur 4.4: Web-side med bilder som lenke, med og uten ramme.

På figuren vises det forrige bildet sammen med bildet uten ramme, slik at det er mulig å sammenligne dem. Begge bildene er en lenke til samme sted, men det går ikke fram av bildet til høyre at det er en lenke. Besøkende til web-siden hvor bildet uten ramme befinner seg, kan derfor ha problemer med å forstå at bildet er en lenke. Hvis du ønsker å bruke bildet uten ramme som lenke, bør du velge alternative måter å forklare besøkende at bildet er en lenke på. Du kan f.eks. la det gå fram av teksten på bildet at det er en lenke, eller du kan la teksten på web-siden henvise til at bildet er en lenke.

4.5.3 Relativ og absolutt adressering

Når du skal legge inn bilder og lenker, må du skrive inn en URL i henholdsvis SRC-attributtet og `href`-attributtet. Denne URL'en kan adresseres absolutt eller relativt.

Ved absolutt adressering skriver du inn hele URL'en til det du vil ha tak i, for eksempel en lenke til "http://stud-vevtjener/~ola/hjemmeside.html". Dette er selvfølgelig den korrekte framgangsmåten når du skal ha tak i en side eller et bilde som ligger på en annen server enn din egen, men det er ganske unødvendig når alle ressursene befinner seg på samme server.

Derfor bør du benytte relativ adressering når du skal referere til ressurser (dokumenter/bilder) som ligger på samme server som du befinner deg på. Det vil si at du adresserer ressursene relativt til dokumentet du arbeider med. Dersom du befinner deg på "`http://stud-vevtjener/~ola/min-side.html`" og skal ha tak i dokumentet som befinner seg på URL'en "http://stud-vevtjener/~kari/teknisk/fagord.html", trenger du ikke skrive inn "`http://stud-vevtjener/`" i `href`-attributtet til lenken. I stedet adresserer du lenken relativt til dokumentet du befinner deg i og bruker den forkortede URL'en /~kari/teknisk/fagord.html ".

For å kunne forstå og benytte relativ adressering er det viktig at du har forståelse for hvordan man manøvrerer mellom kataloger/mapper. Dersom du har arbeidet med MS-DOS eller Unix, vil temaet være kjent for deg, men dersom du er vant til mapper i Windows, vil temaet muligens være vanskelig å forstå, og begreper som "rotkatalog" og "../" kan være ukjente. Derfor skal vi se nærmere på disse begrepene.

4.5.4 Rotkatalog

Når filer er organisert i kataloger (også kalt "mapper"), er det vanlig å se på organiseringen som et hierarki. Nedover i hierarkiet kan man ha underkataloger på mange nivåer. *Rotkatalogen* er toppen av hierarkiet, og vi kaller den for "roten".

På WWW vil rotkatalogen i våre eksempler være den ressursen som er tilgjengelig med URL'en "http://stud-vevtjener/". For eksempel er URL'en til hjemmesiden din på studenttjeneren: "`http://stud-vevtjener/~brukernavn/`". Rotkatalogen student-tjeneren er da "`http://stud-vevtjener/`". Dersom du har lastet opp web-sidene dine til en annen server, er selvsagt rotkatalogen en annen. URL'en til rotkatalogen kan erstattes av en "`/`" (skråstrek) når du skal lenke til dokumenter med relativ adressering.

Eksempel:

> Du har siden din på "`http://stud-vevtjener/min-katalog/min-side.html`" og ønsker å lenke til "`http://stud-vevtjener/olas-katalog/olas-side.html`".

Ettersom begge sidene befinner seg på " stud-vevtjener ", er det unødvendig å skrive inn "http://stud-vevtjener/". I stedet benytter du deg av den egenskapen at "/" tilsvarer "http://stud-vevtjener/". Resultatet blir at du i lenken bruker URL'en "/olas-katalog/olas-side.html".

4.5.5 Gjeldende katalog

Med gjeldende katalog mener vi katalogen du for øyeblikket befinner deg i, det vil si den katalogen der web-siden du arbeider med ligger lagret. Det er også mulig å bruke relativ adressering for å lenke til kataloger som befinner seg høyere og lavere i hierarkiet enn den man arbeider i. Med ordet "høyere" mener vi at man arbeider seg oppover i hierarkiet, mot rotkatalogen, og med "lavere" mener vi nedover i hierarkiet. "Lavere" betyr dermed inn i kataloger under den du arbeider i.

Først skal vi ta for oss hvordan du får tilgang til kataloger lavere enn den du for øyeblikket arbeider i. Det vil du få bruk for ofte, ettersom bruken av underkataloger gir deg mulighet til å systematisere innholdet i en katalog. Ved å dele filene inn i underkataloger kan du sortere dem etter emner og lettere holde oversikten. På den måten vil du skape ditt eget hierarki av filer. For eksempel vil det være vanskelig å holde orden på 125 filer i samme katalog, mens en inndeling i fem underkataloger og 25 filer i hver katalog er mer overkommelig.

Ved relativ adressering vil en underkatalog være tilgjengelig ved å bruke URL'en "katalognavn/". "Katalognavn" er navnet på katalogen, og det er lagt til en "/" (skråstrek) på slutten for å markere at det er en katalog.

Eksempel:

Du har siden din på "http://stud-vevtjener/~brukernavn/webside.html". Tenk deg at du i en underkatalog har liggende ukeplaner for arbeidet ditt, og du ønsker å lenke fra "webside.html" til filen "uke32.html", som befinner seg i katalogen "ukeplaner". Dersom du skulle brukt den absolutte URL'en, ville den være "http://stud-vevtjener/~brukernavn/ukeplaner/uke32.html". I stedet ønsker du å bruke relativ adressering. For å få tilgang til underkatalogen som heter "ukeplaner", bruker du URL'en "ukeplaner/" ved relativ adressering. Du ønsker å lenke til filen "uke32.html", og URL'en du bruker i lenken din, blir da "ukeplaner/uke32.html".

Du er ikke begrenset til å bevege deg kun én katalog ned i hierarkiet. Det er like enkelt å bevege seg flere kataloger nedover. Du legger da bare til et nytt katalognavn og en skråstrek for hver katalog. For eksempel "katalog_1/katalog_2/katalog_3/", som vil gi deg tilgang til dokumentene i katalog_3, som ligger under katalog_2, som igjen ligger under katalog_1.

Det er også mulig å bevege seg oppover i hierarkiet, fra en katalog til katalogen over. Til dette benytter man en spesiell katalogbetegnelse; "../" (punktum-punktum-skråstrek). Denne betegnelsen betyr at du ønsker å få tilgang til dokumentene i katalogen ett hakk over den du for øyeblikket befinner deg i.

Eksempel:

Vi benytter oss av URL'en i forrige eksempel, "http://stud-vevtjener/~brukernavn/ukeplan/uke32.html". Du jobber med dette dokumentet og ønsker å få tilgang til dokumentet "webside.html", som ligger i katalogen over den "uke32.html" befinner seg i. Først må du få tilgang til

katalogen over, og du benytter deg derfor av at "`../`" gir deg tilgang til denne. URL'en du bruker er derfor så langt "`../`". Det er ikke katalogen du ønsker å lenke til, men et spesifikt dokument, og derfor legger du filnavnet til den eksisterende URL'en. Resultatet er at URL'en du bruker blir "`../webside.html`".

Igjen er det mulig å bevege seg oppover mer enn ett nivå i hierarkiet. For å bevege deg to kataloger opp bruker du "`../../`", tre kataloger opp "`../../../`" osv. Du kan også bevege deg opp, for deretter å bevege deg nedover igjen, som for eksempel med "`../../katalog_1/katalog_2/`". Denne URL'en vil gi deg tilgang til katalog_2, som ligger under katalog_1, som igjen ligger to kataloger over den du befinner deg i.

Ved første øyekast virker dette sikkert komplisert, men med litt trening blir det en lek å benytte seg av det. Du syns kanskje det virker komplisert når vi forklarer det? Et tips kan være å tegne opp hierarkiet og bevegelsene som gjøres i eksemplene, og dermed lettere se hva som skjer. Å bruke relativ adressering er svært praktisk, ettersom du da vil slippe å skrive inn lange, kompliserte URL'er hele tiden.

4.6 Meta - taggen

I kapittel 2 fortalte vi at det ved siden av tittelelementet finnes flere elementer du kan bruke i dokumentets hode. Et av disse er meta-elementet, som vi bruker til å fortelle noe om dokumentet, såkalt meta-informasjon.

Meta-elementet har en start-tagg, `<meta>`, men ingen slutt-tagg. Elementet plasseres som sagt i dokumentets hode, og er derfor verken blokknivå- eller tekstnivåelement, ettersom disse begrepene kun brukes om elementer som benyttes i dokumentets kropp.

For å kunne bruke meta-elementet trenger du to attributter: `name=""` og `content=""`. Disse to attributtene kombineres med `<meta>` etter reglene du lærte i studieenhet 1. Hvordan disse to attributtene virker sammen med `<meta>`-taggen er vanskelig å forklare, og vi skal i stedet illustrere det med et eksempel:

```
<meta name="Author" content="Jarle H&aring;vik">
```

Attributtet `name=""` skal fortelle hvilken egenskap ved dokumentet du vil fortelle noe om. Deretter skal attributtet `content=""` inneholde det du ønsker å fortelle om den egenskapen. I vårt tilfelle er det forfatteren av dokumentet ("author" av det engelske ordet for forfatter) vi ønsker å si noe om, og det settes inn i `name`-attributtet. Da er det naturlig at den tilsvarende `content`-attributtet inneholder navnet på forfatteren. I dokumentene du lager, kan du derfor fortelle besøkende at du er forfatteren ved å legge inn denne informasjonen ved hjelp av `meta`-taggen i dokumentets hode.

I tillegg til navn på forfatter er det to andre egenskaper ved dokumenter det er vanlig å benytte meta-elementet til: en beskrivelse av innholdet og nøkkelord som søkemotorer kan bruke. Når du jobber med et stort antall sider og ønsker at de skal legges inn i søkeindekser, vil bruken av meta-elementet gjøre at du slipper å oppgi beskrivelsen og hvilke nøkkelord man skal søke på for hver gang. I stedet kan du legge inn dette ved hjelp av meta-elementet, og dokumentets hode kan se slik ut:

```
<head>
<title>Hannes fysikkside</title>
<meta name="Description" content="Tar for seg det grunnleggende
innenfor 2FY med vekt p&aring; Newtons 3 lover og anvendelse av disse."
< meta name="Keywords" content="newton,2fy,fysikk,kraft,motkraft">
</head>
```

Dersom brukere av WWW søker på "fysikk 2fy" i for eksempel Alta Vista, vil det komme et treff på Hannes fysikkside ettersom ordene det søkes etter passer med nøkkelordene, og beskrivelsen av web-siden som er lagt inn i dokumentets hode, vil komme opp som forklaring på hva siden inneholder.

5. HTML 4.0

5.1 Mål

Etter å ha gjennomgått dette kapittelet, skal du

- ha fått en innføring i HTML 4.0
- vite om hvilken støtte HTML 4.0 har for språk og skriveretning
- kunne lage tabeller med den nye versjonens muligheter
- vite hvordan objekter i websider fungerer
- kunne lage sider som bruker rammer ("frames")

5.2 HTML versjon 4.0

HTML 4.0 ble utgitt av W3C 24. april 1998. Mens versjon 3.2 var forholdsvis liten og lett oversiktlig, har versjon 4.0 blitt atskillig større, og kan derfor skremme folk fra å sette seg inn i den. Dersom man har grunnleggende kunnskaper om versjon 3.2, er dog ikke versjon 4.0 uoverkommelig i størrelse. Det er i hovedsak snakk om tillegg/endringer i forhold til versjon 3.2.

Forskjellene mellom versjon 3.2 og 4.0 kan kort oppsummeres slik:

- Det er lagt inn bedre støtte for språk, både med tanke på hvilke tegn man kan få lagt inn i dokumenter, samt hvilken retning teksten skrives (fra høyre mot venstre).
- HTML 4.0 har utvidet støtte for bruk av script (JavaScript)
- Tabellene har fått flere muligheter, mye med tanke på gruppering av celler og rader
- Man har fått `<OBJECT>` som skal være et felles element for bilder, lyd og Java-applets.
- Rammer ("frames") er lagt inn, og er dermed ansett som tillatt
- Alt som har med layout å gjøre (utseende på tekst, plassering, osv) er uønsket, isteden vil man går over til stilsett (Cascading style Sheets).

5.3 Mer om tabeller

I kapittel 3 lærte vi om tabeller og hvordan de bygges opp. Vi tok ikke for oss annet enn de helt enkleste byggeklossene for tabellene. HTML, i både versjon 3.2 og 4.0 inneholder store muligheter for å manipulere tabeller. Disse mulighetene ble forbigått i stillhet i kapittel 2 ettersom de i stor grad påvirker utseendet til en tabell.

5.3.1 Overskrift og oppsummering

Tabeller har gjerne en overskrift, og i HTML 4.0 har man også lagt inn muligheten for å gi en kort oppsummering av tabellens innehold. Oppsummeringen kan for eksempel leses opp for de som bruker tale-syntese og dermed gi de en bedre forståelse for hva tabellen inneholder. Det er ofte problematisk å gi tabeller et godt utseende i annet enn grafiske nettlesere, noe oppsummeringen er til for å rette på.

Overskrift

Til å sette en overskrift (eller underskrift) til tabellen har man elementet `<caption>`. Start-taggen er allerede gitt, slutt-taggen er `</caption>`.

La oss se på et eksempel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
    <title>Tabeller - overskrift </title>
  </head>
  <body>
    <table border>
      <caption>Oversikt over kaffe-ønsker</caption>
      <tr><th>Navn</th><th>Avdeling</th><th>Melk</th><th>Sukker</th><tr>
      <tr><td>Per Hansen</td><td>Økonomi</td><td>Nei</td><td>Nei</td></tr>
      <tr><td>Line Olsen</td><td>Salg</td><td>Ja</td><td>Nei</td></tr>
      <tr><td>Stein Svingen</td><td>Lager</td><td>Nei</td><td>Ja</td></tr>
    </table>
  </body>
</html>
```

Skript0501.html: Tabell med overskrift - caption

Skriv inn koden over og arkiver den i en ny mappe `kapittel05` med filnavnet `skript0501.html`. Legg også merke til at deklarasjonen nå er endret siden det nå er HTML 4.0 spesifikasjonen som skal benyttes. Videre velger vi nå heretter å legge inne meta-taggen som sier hvem som har forfattet skriptet.

Når vi så setter dette inn i en web-side, kan resultatet se slik ut:

Figur 5.1: Tabell med overskrift

Som vi ser kommer overskriften på toppen av tabellen, og også utenfor tabellens ramme. Vi har også mulighet til å bestemme posisjonen til overskriften, for eksempel gjøre den til en underskrift ved å sette inn `align`-attributtet. Dette er et attributt som det i HTML 4.0 er ønsket skal håndteres med stilsett.

```
<body>
  <table border>
    <caption align="bottom" >Oversikt over kaffe-ønsker</caption>
    <tr><th>Navn</th><th>Avdeling</th><th>Melk</th><th>Sukker</th><tr>
    <tr><td>Per Hansen</td><td>Økonomi</td><td>Nei</td><td>Nei</td></tr>
    <tr><td>Line Olsen</td><td>Salg</td><td>Ja</td><td>Nei</td></tr>
    <tr><td>Stein Svingen</td><td>Lager</td><td>Nei</td><td>Ja</td></tr>
  </table>
</body>
```

Skript0501b.html: Tabell med underskrift - caption

Overskriften vil da komme på undersiden av tabellen, slik:

Figur 5.2: Bilde av tabell med underskrift.

Oppsummering

Oppsummeringen er ikke et eget element, men et attributt som legges inn i `<table>`. Attributtet er `summary=""`, og i den skriver man en kort oppsummering av innholdet i tabellen. Eksempelvis:

```
<body>
  <table border
      summary="Oversikt over kaffeønsker i Bilbruk A/S. Navn, avdeling og
          deres respektive ønsker angående melk og sukker i kaffen.">
    <caption align="bottom" >Oversikt over kaffe-ønsker</caption>
      <tr><th>Navn</th><th>Avdeling</th><th>Melk</th><th>Sukker</th><tr>
      <tr><td>Per Hansen</td><td>Økonomi</td><td>Nei</td><td>Nei</td></tr>
      <tr><td>Line Olsen</td><td>Salg</td><td>Ja</td><td>Nei</td></tr>
      <tr><td>Stein Svingen</td><td>Lager</td><td>Nei</td><td>Ja</td></tr>
  </table>
</body>
```

Skript0501c.html: Tabell med oppsummering - summary

Denne oppsummeringen vil ikke vises i en grafisk nettleser, men kan leses opp dersom brukeren har tale-syntese. Eventuelt kan den også vises i nettleser som "pop-up help" når brukeren legger muspekeren over tabellen.

5.3.2 Sammenkobling av celler - colspan og rowspan

Ofte ønsker man å koble sammen en eller flere celler, og dette er også mulig. Man kan koble sammen celler over to eller flere rader, eller to eller flere kolonner om man ønsker det.

Vi kan først se på et eksempel hvor vi kobler sammen fire celler for å lage en overskrift som går over fire kolonner. Til å gjøre dette bruker vi attributtet `colspan=""`. Attributtet tar et tall som parameter, og det talles forteller hvor mange kolonner cellen skal gå over. Koden ser slik ut:

```
<body>
   <table border>
      <tr><th colspan="4">Overskrift</th></tr>
      <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
   </table>
<body>
```

Skript0502.html: Tabell med sammenkobling av kolonner

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502.html`.
Når vi så setter dette inn i en web-side, kan resultatet se slik ut:

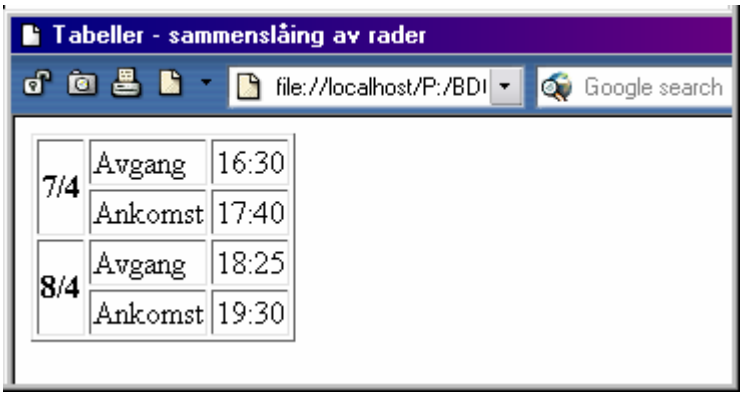
Figur 5.3: Eksempel på tabell med sammenkoblede celler over 4 kolonner .

På samme måte kan vi koble sammen celler over to eller flere rader med attributtet `rowspan =""`. Dersom vi for eksempel har en ruteplan, og denne skal ha både avgangs- og ankomst-tider kan vi løse dette med følgende tabell:

```
<table border>
  <tr><th rowspan="2">7/4</th><td>Avgang</td><td>16:30</td></tr>
  <tr><td>Ankomst</td><td>17:40</td></tr>
  <tr><th rowspan="2">8/4</th><td>Avgang</td><td>18:25</td></tr>
  <tr><td>Ankomst</td><td>19:30</td></tr>
</table>
```

Skript0502b.html: Tabell med sammenkobling av kolonner

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502b.html`.
Når vi så setter dette inn i en web-side, kan resultatet se slik ut:



Figur 5.4: Eksempel på tabell med sammenslåtte celler over 2 rader

4.3.3 Tabell-utseende

Vi har også gode muligheter for å påvirke tabellens utseende med tanke på rammen rundt, avstanden mellom cellene og "luft" rundt teksten i en celle. I motsetning til `colspan=""` og rowspan="" som behandler en og en celle, vil endringene av de nevnte ting endre hele tabellen. La oss se på de forskjellige mulighetene:

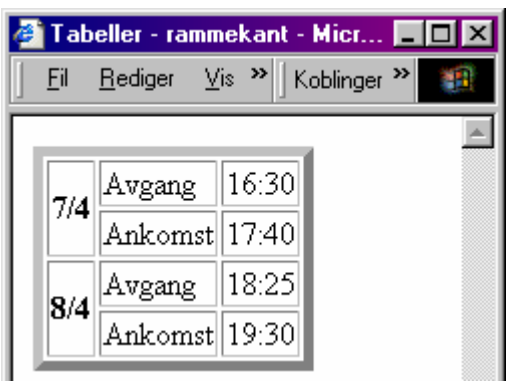
Ramme

Vi kan starte med rammen rundt tabellen, noe vi allerede har benyttet oss av. Denne rammen kan enten spesifiserer som på eller av, eller man kan si at den skal ha en viss bredde i punkter på skjermen (pixler). Ønsker man ikke en ramme i tabellen, bruker man `<table>`, og ønsker man en standard-ramme definert av nettleser, bruker man `<table border>` som nevnt i kapittel 3. Vi kan også definere rammen i pixler, og da setter man inn verdien ved hjelp av `border=""`, slik:

```
<table border="5">
  <tr><th rowspan="2">7/4</th><td>Avgang</td><td>16:30</td></tr>
  <tr><td>Ankomst</td><td>17:40</td></tr>
  <tr><th rowspan="2">8/4</th><td>Avgang</td><td>18:25</td></tr>
  <tr><td>Ankomst</td><td>19:30</td></tr>
</table>
```

Skript0502c.html: Tabell med sammenkobling av kolonner og ramme

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502c.html`. Resultatet i nettleseren kan se slik ut (dette vil variere fra nettleser til nettleser):



Figur 5.5: Tabell med ramme, slik den vises i nettleser.

Legg merke til hvorledes avstanden mellom tabell-cellene og luften i selve cellene fortsatt er lik, mens rammen rundt med en gang ble større.

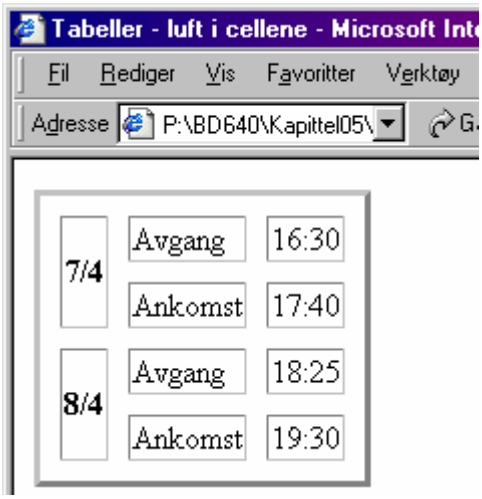
Vi kan også endre avstanden mellom cellene. Dette gjøres med attributtet `cellspacing=""`.

La oss gjøre rammen litt mindre, og istedet lage større avstand mellom cellene:

```
<table border="3" cellspacing="5" >
  <tr><th rowspan="2">7/4</th><td>Avgang</td><td>16:30</td></tr>
  <tr><td>Ankomst</td><td>17:40</td></tr>
  <tr><th rowspan="2">8/4</th><td>Avgang</td><td>18:25</td></tr>
  <tr><td>Ankomst</td><td>19:30</td></tr>
</table>
```

Skript0502d.html: Tabell med større avstand mellom cellene.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502d.html`. Resultatet i nettleseren kan se slik ut :



Figur 5.6: Tabell med økt avstand mellom cellene.

Og til slutt kan vi sette avstanden mellom cellene litt mindre og i stedet gi cellene mere luft. Dette gjør vi ved hjelp av attributtet cellpadding="" som bestemmer avstanden fra innholdet av en celle og til kanten av cellen.

```
<table border="3" cellspacing="2" cellpadding="5" >
  <tr><th rowspan="2">7/4</th><td>Avgang</td><td>16:30</td></tr>
  <tr><td>Ankomst</td><td>17:40</td></tr>
  <tr><th rowspan="2">8/4</th><td>Avgang</td><td>18:25</td></tr>
  <tr><td>Ankomst</td><td>19:30</td></tr>
</table>
```

Skript0502e.html: Tabell med større avstand mellom cellene.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502e.html`.

Resultatet i nettleseren kan se slik ut :

Figur 5.7: Tabell med mer luft i cellene

4.3.4 Tabellbredde og cellebredde

HTML gir oss også mulighet til å sette bredde på tabellen og cellene. Setter man bredde på tabellen gjelder dette for hele tabellens bredde fra venstre kant til høyre kant. En gitt bredde på cellene gjelder da kun for den kolonnen cellen er i.

Tabellens bredde kan enten oppgis i punkter på skjermen (pixler) eller i prosent. Dersom man bruker prosent, er det antall prosent av tilgjengelig bredde i nettleser som brukes. Å sette tabell-bredden i pixler er en uting, ettersom man ikke vet hvilken bredde brukerens nettleser har, og en for bred tabell vil da føre til at han må "scrolle" (flytte skjermbildet) for å se hva tabellene inneholder. Dersom det brukes prosentvis størrelse, vil tabellen endre størrelse avhengig av nettleser-vinduet.

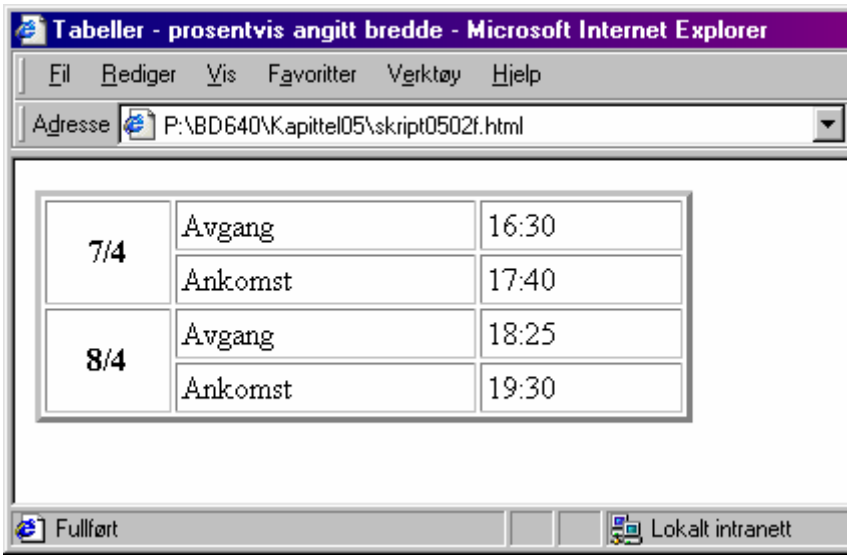
Vi kan endre på bredden til tabellen vi brukte i forrige eksempel. Til dette bruker vi attributtet `width=""`.

```
<table border="3" cellspacing="2" cellpadding="2" width="75%" >
  <tr><th rowspan="2">7/4</th><td>Avgang</td><td>16:30</td></tr>
  <tr><td>Ankomst</td><td>17:40</td></tr>
  <tr><th rowspan="2">8/4</th><td>Avgang</td><td>18:25</td></tr>
  <tr><td>Ankomst</td><td>19:30</td></tr>
</table>
```

Skript0502f.html: Tabell med prosentvis angitt bredde.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0502f.html`.

Resultatet i nettleseren kan se slik ut :



Figur 5.7: Tabell med prosentvis angitt bredde (75%).

Dersom vi gjør nettleservinduet større, vil tabellen også bli bredere, og vise versa. For en liten tabell som denne er fastset-telse av størrelsen ikke spesielt nød-vendig, men det kan være nyttig i andre sammenhenger.

Når man ønsker å sette bredden på cellene, gir HTML 4.0 oss kun mulighet til å sette denne i antall pixler. De To Store(tm) nettleserne, Netscape Navigator og Internet Explorer, støtter at man også oppgir en prosentvis bredde, men man er uenig i hvorvidt det er snakk om prosent av tabellens bredde, tilgjengelig plass på skjermen eller tilgjengelig plass for tabell-celler etter at andre celler har fått sitt. I praksis vil man sjelden ha behov for å endre bredden på celler, men når behovet først er der er det viktig å vite at det kun er antall pixler man kan sette.

Den største bruken av fast cellebredde er når man bruker tabeller til layout. Det er vanlig å lage en tabell hvor celler på venstre billedkant har en fast bredde, og gjerne til sammen utgjør en meny folk kan bruke til å navigere med. Man lar så en stor celle på høyre side inneholde brødteksten. Dette kapitlet tar ikke sikte på å lære bort layout-triks for WWW, og derfor vil vi ikke komme nærmere inn på dette. Dersom eksempler er ønsket, er alt man trenger å gjøre å besøke noen av de mest besøkte stedene på web. Det vil bli lagt lenker til noen steder som bruker tabeller under "Ressurser.

5.4 Internasjonalisering

5.4.1 Språk

I HTML 4.0 er det lagt inn bedret støtte for språk, både for hele dokumentet og i deler av det. Dette for å hjelpe søkemotorer, brukere med talesyntese, og for at tekst skal vises med korrekte aksenter for språket det gjelder.

Alle elementer i HTML har nå fått støtte for et nytt attributt: `lang=""`. Dette attributtet skal spesifisere hvilket språk som er brukt i teksten elementet inneholder. Det brukes en kode bestående av to bokstaver for å benevne hvilket språk som brukes. Programmeringsspråk eller andre datarelaterte språk er utelukket fra listen over godkjente koder. Dersom du er i tvil om hvilken kode du skal bruke, kan du finne en liste over godkjente koder i ISO 639.

Dersom det er en spesiell dialekt som skal brukes, kan dette legges til med en bindestrek samt navnet på dialekten. La oss se på et eksempel-dokument:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="no">
   <head>
      <meta name="author" content="JARLE.HAVIK@HISF.NO">
      <title>Et flerspråklig dokument</title>
   </head>

   <body>
      ...tekst som følger her blir tolket som norsk språklig...
      <p LANG="en">...tolket som engelsk...</p>
      <p>...tolket som norsk igjen...</p>
      <p>...norsk tekst avbrutt av <em lang="fr">litt fransk</em>,
      mens dette er norsk igjen...
   </body>
</html></HTML>
```

Skript0503.html: Eksempel på angivelse av språk i et HTML-dokument.

Her er det spesifisert språk flere steder. Først benyttes attributtet `lang="no"` satt inn i `<html>` for å spesifisere at dokumentet er norskspråklig. All tekst i dokumentet som ikke har et språk definert blir dermed regnet som norsk.

Vi har et avsnitt som starter med `<p lang="en">`, og dette avsnittet er da på engelsk. Litt senere i dokumentet har vi satt inn noe som er på fransk, og det er brukt `<em lang="fr">`, slik at den fremhevede teksten er definert som fransk.

Hvilket språk som vil bli definert dersom du ikke setter et språk i dokumentet er avhengig av oppsettet til brukeren.

5.4..2 Skriveretning

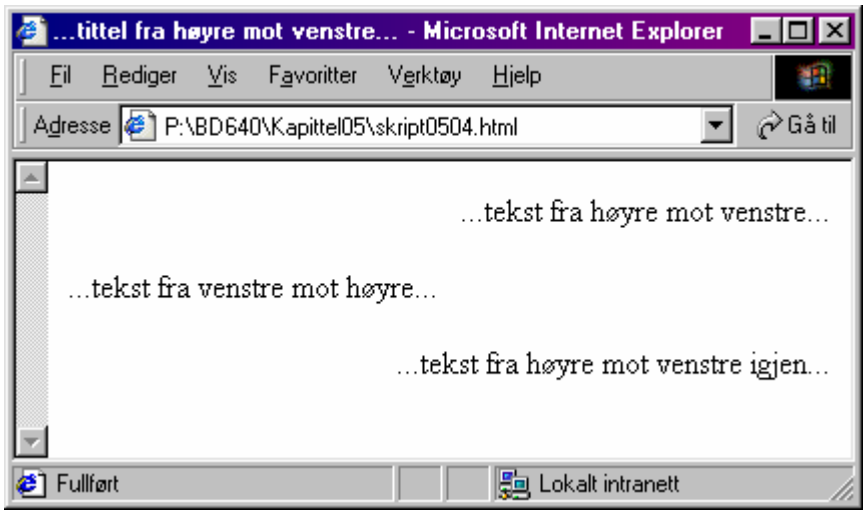
Tidligere HTML-versjoner har ikke lagt vekk på internasjonalisering, og har derfor heller ikke tatt hensyn til hvilken retning et språk skrives. Utgangspunktet har vært fra venstre mot høyre, men det er ikke alle språk som skrives på den måten. I HTML 4.0 har man lagt vekt på internasjonalisering, og det medfører også støtte for at tekst ikke skrives fra venstre mot høyre.

La oss se på et eksempel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
<html dir="rtl">
    <head>
      <title>...tittel fra høyre mot venstre...</title>
    </head>
    <body>
      ...tekst fra høyre mot venstre...
      <p dir="ltr">...tekst fra venstre mot høyre...</p>
      <p>...tekst fra høyre mot venstre igjen...</p>
    </body>
</html>
```

Skript0504.html: Eksempel på endring av skriveretning.

Igjen er det satt inn et attributt i `<html>`, denne gangen er det attributtet `DIR="rtl"`. Det medfører at dokumentet settes opp med tanke på at all tekst skal skrives fra høyre mot venstre. Som det går frem av eksemplet vil dette også gjelde tittelen på dokumentet.



Figur 5.8:Eksempel på skriftretning.

`dir=""` kan også brukes til å endre retning på element-nivå, og det er vist med avsnittet midt i dokumentet. Her brukes `<p dir="ltr">`, slik at teksten blir korrekt vist med retning fra venstre mot høyre.

Standard retning på tekst er definert til å være fra venstre mot høyre.

5.4.3 Tegnsett

Den gang IBM-pc'en ble skapt ga man den et tegnsett som bestod av bare 128 tegn. Det var nok til alfabetet, en del kontrolltegn, samt andre nyttige tegn (parenteser, tall, osv). Etterhvert fikk man behov for å utvide dette tegnsettet, ettersom flere land tok i bruk datamaskiner. IBM løste problemet ved å utvide tegnsettet til 256 tegn. Ved hjelp av programvare kunne man bytte mellom forskjellige tegnsett, avhengig av hvilket språk man skulle vise på skjermen.

Etterhvert slo Microsoft Windows igjennom. Windows brukte ikke samme metoden for å endre språk, men brukte tegnsettet ISO-8859-1 (også kalt ISO Latin-1) som standard for land i den vestlige verden. Hovedproblemet med dette var at tegnsettet fortsatt kun inneholdt 256 tegn, hvilket ikke er nok for å dekke alle språk og varianter man trenger.

I HTML er ISO-8859-1 standard tegnsett for dokumentene. Dette medfører at man har 256 tegn tilgjengelig, og noen av de benytter man entiteter for å kode inn, slik at de ikke vises feil på skjermen. Problemet med andre språk og tegnsett er derfor fortsatt ikke løst.

Løsningen er Unicode. I HTML 4.0 har man gjort det tilgjengelig å kunne bruke entiteter med Unicode, slik at man kan kode inn tegn fra et 16-bits tegnsett. Dette medfører at man har ett tegnsett som dekker alle språk og varianter man trenger.

Forskjellen mellom de normale entitetene og Unicode-entitetene er hvordan de kodes. La oss se på et eksempel:

Det norske tegnet "å" kan enten tastes inn direkte, ettersom det finnes i ISO-8859-1, eller det kan legges inn via en entitet. Den numeriske-entiteten for "å" er å. Tegnet finnes også i Unicode, men har da heksadesimal nummer E5. Den numeriske entiteten med Unicode blir da å.

På samme måten kan man kode inn de andre norske tegnene, eller ethvert annet tegn man har behov for å legge inn. Det man må huske på er at en god del nettlesere ikke har støtte for dette, og at resultatet kan bli at tegnene ikke vises korrekt.

6 Rammer (frames)

Rammer ble introdusert av Netscape når de slapp versjon 2.0 av nettleser Navigator. Noen mener at dette var fordi Netscape ikke orket å legge inn støtte for `<link>`, et element som gir forfattere mulighet til å definere dokumenters relasjon til hverandre. Et eksempel kan være en større samling dokumenter, hvor vi ved hjelp av `<link>` kan fortelle nettleser hvor forrige kapittel er, neste kapittel, samt innholdsfortegnelsen, og deretter kan nettleser gi brukeren navigeringsmuligheter for dette. En slik løsning er delvis implementert i NCSA Mosaic 3.0 for Windows 95 og gjør bruken av rammer overflødig.

Verden ville det nå ikke engang slik at `<link>` skulle bli en standard, noe Netscapes store markedsandel på det tidspunktet de slapp Navigator versjon 2.0 sannsynligvis kan være årsaken til. Etterhvert har det blitt mer og mer akseptert at man bruker rammer, ettersom både Netscape Navigator og Microsoft Internet Explorer nå støtter dette, samt at i HTML 4.0 er det mulig å skrive standardiserte dokumenter som bruker rammer.

6.1 Inndeling av nettleser-vinduet

Ved hjelp av rammer kan man dele inn nettleser-vinduet enten vertikalt eller horisontalt. For eksempel kan vi dele inn vinduet i en liten del til venstre og en stor del til høyre, legge en liste med lenker i venstre del, og la innholdet av disse lenkene komme opp i høyre del. På den måten klarer vi lett å holde menyen med lenker statisk og dermed tilgjengelig uansett hvilken side du besøker. Dette er den store fordelen med rammer, at den gjør det veldig enkelt å ha en meny stående fast på alle sider.

Det er på tide at vi tar en titt på hvorledes et dokument med et rammesett (frameset) ser ut. En ting er viktig å huske på når man arbeider med rammer: Siden du ser på skjermen består av et rammesett som igjen består av selvstendige websider. Dersom du har et rammesett som deler nettleser-vinduet i to, må du derfor ha tre filer; selve rammesettet samt en fil for hver av de to delene. Se på dette eksemplet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
    <title>Et enkelt ramme-eksempel</title>
  </head>
  <frameset cols="20%,80%">
      <frame src="ramme-fil1.html">
      <frame src="ramme-fil2.html">
  </frameset>
</html>
```

Skript0601.html: Eksempel på et enkelt ramme-oppsett

Først i dokumentet kommer deklarasjonen, denne gangen definerer den dokumentet som et rammesett (ved hjelp av "HTML 4.0 Frameset") og i tillegg gir den en URL til hvor dokument-definisjonen for rammesettet befinner seg ("http://www.w3.org/TR/REC-html40/frameset.dtd").

Videre ser vi at dokument-hodet ikke er forandret på noen måte, tittelen til dokumentet legges inn på vanlig måte. Som du kanskje har oppdaget, mangler

dokumentet en kropp. `<body>` og `</body>` er ikke å finne i dette dokumentet. I stedet har man byttet ut `<body>` med elementet `<frameset>`.

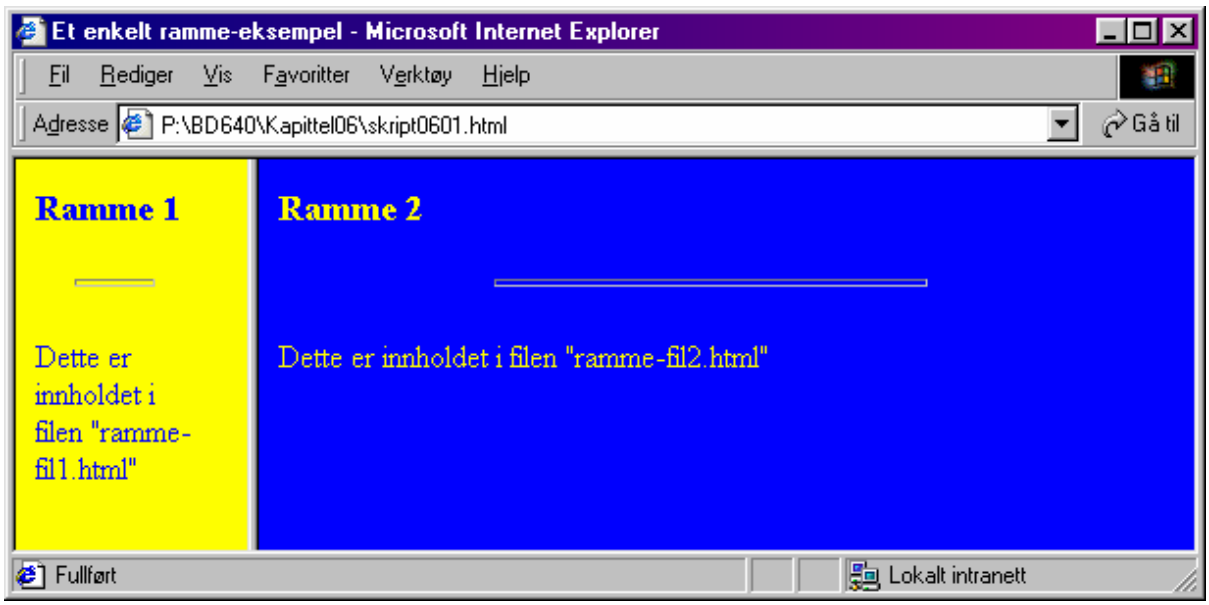
Elementet for rammesettet har dermed start-taggen `<frameset>` og slutt-taggen `</frameset>`. Et rammesett står aldri alene, det vil si du kommer alltid til å legge inn en eller flere andre attributter der. Dette er også vist i eksemplet, hvor vi bruker `cols="20%,80%"`.

`cols=""` deler inn nettleser-vinduet vertikalt, dvs i kolonner. Det er i utgangspunktet ingen begrensning i hvor mange deler man kan ha, men med 4-5 deler begynner ting fort å bli lite oversiktlig. Vanlig er å dele inn i 2 eller 3 deler.

Vi har også et annet attributt for å dele inn nettleser-vinduet, og den tar seg av horisontal inndeling. Attributtet er `rows=""` og virker på samme måte som `cols=""`, men med den lille forskjellen at nettleser-vinduet deles horisontalt. La oss se litt nærmere på hvordan man bruker de.

Legg også merke til at hver ramme, definert med taggen `<frame>` bruker en egen web-side. Hvor siden befinner seg forteller man nettleser ved hjelp av src="", og i vårt tilfelle vil venstre ramme inneholde det filen "ramme-fil1.html" gjør. Høyre ramme vil på samme måte inneholde det "ramme-fil2.html" gjør.

Dette ble kanskje litt høytsvevende, så vi skal se hvordan dette rammesettet ser ut i en nettleser. Slik rammesettet er definert, skal web-siden deles inn i to kolonner. Den venstre rammen skal ha 20 % av bredden på nettleser-vinduet, mens den høyre skal ha de resterende 80 %. Dette kan da se slik ut:



Figur 6.1: Eksempel på rammesett, slik det vises i nettleser.

6.1.1 Parametre til rows="" og cols=""

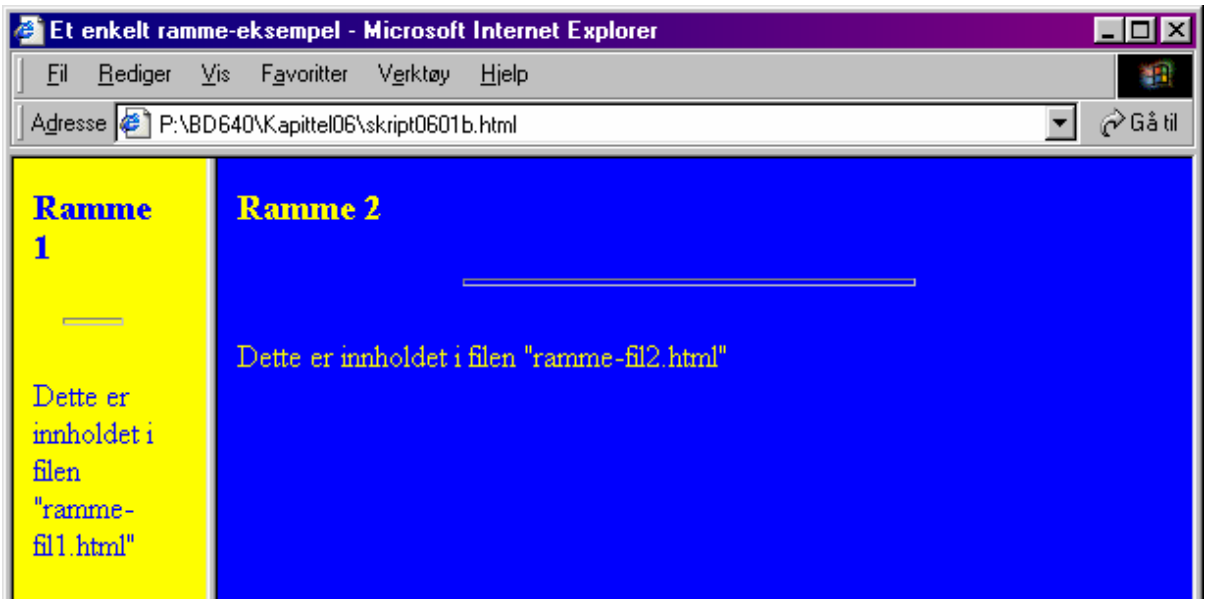
Som sagt benyttes disse for å dele inn nettleser-vinduet. Begge attributtene tar parametre i form av en komma-inndelt liste. Denne komma-indelte listen er det vanlig å la inneholde en eller flere av tre alternativer. Dersom man setter inn et tall direkte vil dette tolkes som en piksel-verdi (punkter på skjermen), og det vil da settes av det nevnte antallet piksler til en ramme. Man kan også sette inn prosentverdier ved å oppgi et tall og '%' etterpå. Rammen(e) vil da bli tilegnet plass

avhengig av størrelsen på nettleser-vinduet. Til sist har man en spesiell verdi, '*', som betyr "det som er igjen". La oss se på noen eksempler:

```
<frameset cols="100,*">
    <frame src="ramme-fil1.html">
    <frame src="ramme-fil2.html">
</frameset>
```

Skript0601b.html: Eksempel på et enkelt ramme-oppsett med pixel angivelse

I dette eksemplet vil det settes av 100 piksler til en ramme til venstre, og det som er igjen av plass i nettleser-vinduet vil bli gitt til en ramme til høyre. Dette kan se slik ut:



Figur 6.2: Ramme-eksempel med venstre ramme 100 piksler bred.

I skript0601.html satte vi rammene til cols="20%,80%". Dette kunne vi også skrevetpå følgende måte:

```
<frameset cols="20%,*">
  <frame src="ramme-fil1.html">
  <frame src="ramme-fil2.html">
</frameset>
```

Her vil nettleser sett av 40 % av nettleser-vinduet til rammen til venstre, og resten av plassen (60 %, logisk nok) får rammen til høyre.

Dette kan selvfølgelig kombineres eller utvides slik man ønsker. Gitt at man vil dele inn siden i 3 deler, med en meny på venstre, og høyre side gjør man det enkelt og greit slik:

```
<frameset cols="100,*,100">
  <frame src="venstre.html">
  <frame src="midt.html">
  <frame src="hogre.html">
</frameset>
```

Skript0601c.html: Eksempel på et tredelt ramme-oppsett med pixel angivelse

Nettleser setter da først av 100 piksler for rammen til høyre, deretter 100 pixler for rammen til venstre, og resten gis til rammen i midten. Legg også merke til hvordan vi her har vært nødt til å fortelle nettleser hvor den finner den tredje filen, ettersom vi har et rammesett bestående av tre rammer. I en nettleser kan rammesettet se slik ut:

Figur 6.3: Rammesett med tre rammer.

6.1.2 Nettlesere som ikke støtter rammer

Rammer er en teknologi som har kommet for å bli, men det betyr ikke automatisk at det er støttet godt av alle nettlesere. Ikke alle brukere ønsker å henge seg på oppgraderingskarusellen og bruker derfor eldre nettlesere. Når man også husker på at en av grunntankene bak WWW var at informasjonen skal være tilgjengelig uansett plattform, betyr det at man må være klar over at man ikke vet hvilken nettleser brukeren har.

Et rammesett inneholder ikke noe informasjon, i stedet er det web-sidene rammesettet peker til som inneholder informasjonen. En nettleser skal være skapt slik at når den treffer på et element den ikke kjenner, dvs noe mellom < og >, skal den allikevel prøve å vise innholdet. Et rammesett har ikke innhold, ergo kan nettleser heller ikke vise noe.

For å løse dette problemet har man elementet `<noframes>`. Innholdet av dette elementet er det en nettleser som ikke støtter rammer skal vise. Vi kan ta utgangspunkt i et av de tidligere eksemplene og utvide det litt:

```
<frameset rows="40,*">
    <frame src="toc.html">
    <frame src="main.html">
</frameset>
```

Dette rammesettet lager en ramme på skjermen som er 40 piksler høy, og ettersom web-siden den inneholder heter "toc.html" er den ment å vise en innholdfortegnelse (eng: Table Of Contents). Resten av nettleser-vinduet skal vise innholdet av dokumentene innholdsfortegnelsen lenker til. Eksempelvis kan nettleser-vinduet se slik ut:

Figur 6.4: Rammesett med menylinje øverst.

En nettleser som ikke støtter rammer vil ikke ha noe innhold å vise frem, så derfor kan vi lage en forklaring for de ved hjelp av `<noframes>`. Eksemplet vårt blir da:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
  <title>Rammesett med to rammer</title>
  </head>
  <frameset rows="40,*">
    <frame src="toc.html">
    <frame src="main.html">
    <noframes>
       <h1>Norske aviser</h1>
       <ul>
           <li><a href="http://www.vg.no">Verdens Gang</a></li>
           <li><a href="http://www.dagbladet.no">Dagbladet</a></li>
           <li><a href="http://www.aftenposten.no">Aftenposten</a></li>
       </ul>
     </noframes>
  </frameset>
</html>
```

Skript0602.html: Eksempel på rammesett som også tar høyde for nettlesere som ikke støtter rammer

Eksemplet vårt kan da se slik ut, når vi ser på det med en nettleser som ikke støtter rammer:

Figur 6.5: En web-side med innhold i <noframes>.

I praksis vil det som står mellom <noframes> og den avsluttende </noframes> være mer eller mindre det samme som innholdet av "toc.html", og på den måten la de som ikke har støtte for rammer få tilgang til innholdsfortegnelsen og dermed alle dokumentene.

En viktig ting å huske på er å ha et meningsfylt innhold i <noframes>. Det er et utall av websider tilgjengelig som kun forteller brukeren at nettleser hans ikke støtter rammer, og at han burde skaffe seg en "bedre" nettleser (avhengig av hva man mener er "bedre"). En slik melding er noe man burde unngå, ettersom mange oppfatter det som ganske frekt at noen andre forteller dem hvilken nettleser som er best.

Dersom du nå har skrevet inn og prøver å kjøre skript0602d.html vil du oppdage at lenkene til de tre avisene ikke fungerer. Dette er helt som det skal være. Vi kommer tilbake til hva som gjøres for at dette skal virke etterhvert.

6.2 Avansert bruk av rammer

Så langt har vi holdt oss til å dele inn nettleser-vinduet enten vertikalt eller horisontalt. Dette er forholdsvis enkelt å forholde seg til, og med litt trening går det lett. Nå skal vi se på noen mer avanserte eksempler, blant annet hvordan man nøster rammesettene, lager lenker som oppdateres i andre rammer, samt fjerner eventuelle mellomrom mellom rammene ("frameborder").

6.2.1 Nøsting av rammesett

Rammesett kan nøstes, på samme måte som lister kan nøstes. Man kan på den måten dele inn skjermen slik man ønsker, både horisontalt og vertikalt.

Første eksempel skal vi se på et rammesett som følger etter følgende retningslinjer:

- Skjermbildet deles først i to vertikalt, en venstre og en høyre del
- Høyre del av nettleser-vinduet skal være stort og åpent for å vise fram tekst
- Venstre del skal så igjen deles inn i tre deler horisontalt.
- Øverst i venstre hjørne skal brukeren gis muligheten til å velge språk, norsk eller engelsk
- Nederst i venstre hjørne er det en lenk tilbake til hovedsiden
- Ved valg av språk kommer det opp en liste over tilgjengelige filer, listen vises i midt-rammen på venstre siden.

Nettleser-vinduet skal derfor deles inn i fire deler. På grunn av retningslinjene som skal gjelde for nettsiden er det kanskje lett å forstå måten problemet løses på, ved at man først deler nettleser-vinduet i to, og deretter deler venstre side i tre. Rammesettet ble da slik:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
```

```
    <title>Eksempel på nøstede rammer</title>
  </head>

  <frameset cols="30%,*">
    <frameset rows="35,*,50">
      <frame src="sprakvalg.html">
      <frame src="liste.html" name="liste">
      <frame src="hjem.html">
    </frameset>
    <frame src="Hovedside.html" name="hoved">
  </frameset>

</html>
```

Skript0603.html: Eksempel på nøsting av rammer

Det første vi gjør er å dele skjermbildet i en venstre og en høyre del. Venstre del gis 33 % av bredden på nettleser-vinduet, resten gis til høyre delen. Dersom vi hadde fulgt de tidligere eksemplene, ville en også hatt en `<fram src="...">`, men ettersom vi nå ønsker å dele inn venstre side ytterligere, må vi nå opprette enda et rammesett. Dette rammesettet deler venstre del inn i tre nye deler. Øverst har vi en ramme som er 35 piksler høy, nederst har vi et område som er 50 piksler høy, og resten gis til området i midten.

Etter at rammesettet som definerer venstre del er avsluttet, følger rammen som er til høyre med referansen til web-siden den skal inneholde. Eksemplet vil i en nettleser kunne se slik ut:

Figur 6.6: Eksempel på nøstede rammer.

6.2.2 Lenker mellom rammer

Du la kanskje merke til bruken av attributtet **name**="" i det forrige eksemplet? Veldig ofte når man jobber med rammer, ønsker man at lenker i en ramme skal åpnes i en annen ramme. Et eksempel er å ha en innholdsfortegnelse i venstre ramme, og la dokumentet åpnes i høyre ramme. For å få til dette er det to kriterier som må være tilfredsstilt:

3 Rammen man ønsker lenken åpnet i må være navngitt med attributtet **name**=""

3 Lenken må definerer rammen som mål med attributtet **target**=""

Bruken av name="" ble vist i forrige eksempel, hvor vi ga to av rammene navn, henholdsvis "liste" (på venstre side) og "hoved" (på høyre side).

Det man så gjør er å bruke target="" i kombinasjon med en allerede eksisterende lenk for å fortelle nettleser at lenken skal åpnes i rammen, hvis navn korresponderer med navnet som settes inn i target="". Sett at vi har en lenke i en av rammene våre, og vil at denne lenken skal åpnes i rammen som vi har kalt "hoved". Lenken vi da må lage må da se slik ut:

```
<A HREF="foo.html" TARGET="hoved">
```

Vi ber da nettleser om å åpne web-siden "foo.html" i rammen som heter "hoved".

Vi kan nå ta frem skript0602.html og gjøre følgende endringer:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
            "http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
  <title>Eksempel på lenking mellom rammer</title>
  </head>
  <frameset rows="40,*">
    <frame src="toc.html" name="avismeny">
  <frame src="main.html" name="hoved">
  <noframes>
    <h1>Norske aviser</h1>
    <ul>
      <li><a href="http://www.vg.no">Verdens Gang</a></li>
      <li><a href="http://www.dagbladet.no">Dagbladet</a></li>
      <li><a href="http://www.aftenposten.no">Aftenposten</a></li>
    </ul>
  </noframes>
  </frameset>
</html>
```

Skript0602b.html: Eksempel på navngivning av rammer

For å se hvordan dette brukes må vi også se på innholdet i toc.html:

```
<html>
  <head>
    <meta name="author" content="JARLE.HAVIK@HISF.NO">
  </head>
  <body bgcolor="#ffff99" text="#000000">
    <a href="http://www.vg.no" target="hoved">Verdens Gang</a> 
    <a href="http://www.dagbladet.no" target="hoved">Dagbladet</a> 
    <a href="http://www.aftenposten.no" target="hoved">Aftenposten</a> 
  </body>
</html>
```

toc.html: Eksempel på html-fil med lenker til navngitte rammer

Legg også merke til hvordan mellomrom mellom lenkene oppnås ved å benytte entiteten   - denne gir et enkelt mellomrom.

Bortsett fra navnet på en ramme har man også muligheten til å bruke enkelte spesielle navn, hvorav disse to stort sett er de eneste som brukes:

_blank

Medfører at lenken åpnes i et nytt nettleser-vindu som ikke er navngitt

_top

Nettleser fjerner hele rammesettet og åpner lenken i et tomt nettleser-vindu. Dette er altså måten man får fjernet rammesettet, men lenken blir fortsatt åpnet i samme vinduet som brukeren står i.

En annen egenskap det kan være verd å nevne er muligheten til å bruke et navn i `target=""` som ikke eksisterer, dvs at rammen ikke er definert. Det vil medføre at nettleser åpner et nytt vindu, og det vinduet vil da være navngitt. Dersom samme `target=""` brukes igjen, vil lenkene åpnes i det navngittte nettleser-vinduet.

6.2.3 Hvordan fjerne mellomrom mellom rammene?

I utgangspunktet vil de fleste nettlesere lage et lite mellomrom mellom rammene. Iblant benytter man rammer for å skape layout, og ønsker derfor ikke at dette mellomrommet skal være der. I stedet ønsker man at alle rammene skal være tett inntil hverandre, hvilket også er mulig. Ifølge HTML 4.0 løses problemet ved å legge til et nytt attributt for hver ramme, `FRAMEBORDER="0"`. Koden for *en ramme* blir da:

```
<frame src="foo.html" frameborder="0">
```

Dessverre er det slik at noen få nettlesere ikke støtter HTML 4.0. Derfor har man en liten snarvei for å sørge for at ting faktisk blir som man vil, selv om det ikke er helt ifølge standarden. Løsningen er å definere *rammesettet* slik:

```
<frameset ... border="0" frameborder="0" framespacing="0">
```

6.3 Ulempene med rammer

Rammer har enkelte fordeler, men det er også en god del ulemper man må ta hensyn til når man jobber med de. Ett av hovedproblemene med rammer er at den bryter et av prinsippene Tim Berners-Lee hadde da han skapte WWW: Den siden du ser på skjermen har en gitt URL. Ved hjelp av denne URL'en kan man da bokmerke siden, eller lenke til den. Rammer bryter dette prinsippet med en gang du følger en lenk innover et rammesett. Da er ikke lenger den URL'en du har lik den siden du ser på skjermen. Resultatet er at rammesettet du da ser ikke kan bokmerkes eller lenkes til. Du kan lenke til enkeltstående dokumenter i et rammesett, men da mister du konteksten dokumentet befant seg i (det vil si det omkringliggende rammesettet).

Dette problemet med at en side ikke kan representeres med en gitt URL fører også til problemer når siden skal indekseres for å legges inn i et av de kjente søkestedene på nettet. Søkeroboten har liten forståelse for rammer, men kan ende opp med å indeksere enkeltstående dokumenter. Igjen mister man kontekst når man søker etter dokumentet og følger lenkene man får som resultat av søket.

Rammer har også en tendens til å legge vekt på layout istedenfor struktur. Som nevnt i kapittel 1 er HTML i utgangspunktet ment å definere strukturen i et dokument. Det er ikke dermed sagt at layout er noe man ikke skal tenke på, men

rammer legger mer vekt på layout enn andre ting. En forfatter kan dermed få problemer, ettersom denne fokuseringen på layout kan følge resten av web-sidene han skaper.

Å skrive ut et rammesett er også et kjent problem. Skal man få det til med noenlunde hell, må man åpne hver ramme i et enkelt nettleservindu og skrive ut innholdet av hver ramme for seg. Det er ikke praktisk mulig å skrive ut et rammesett på en forståelig måte.

Rammer vil også medføre dobbelt arbeid for forfatteren, ettersom han må skape en side som bruker rammer og en side for de som ikke bruker rammer. Sistnevnte legges da inn ved hjelp av `<noframes>`. Dersom han i stedet sørger for at den siden som ikke bruker rammer er godt laget, vil han slippe dette problemet.

Et annet problem merker brukerne mest til, det at med en gang man har et rammesett forsvinner noe av plassen i nettleser-vinduet. Dersom man har en ramme med innholdsfortegnelse, en med en overskrift og en siste ramme for selve teksten på siden, vil den siste rammen ha adskillig mindre plass tilgjengelig enn hva som kunne ha vært mulig uten bruk av rammer. Det samme gjelder da lenker innover i rammesettet som også da vil ha samme begrensningen. Et kjent problem med HTML er at du ikke har noen anelse om hvilket oppsett brukeren som ser på siden har. Det er derfor ikke spesielt hyggelig mot brukerne å la en god del av plassen på web-sidene gå bort til ting han/hun ikke finner spesielt nyttig.

7 Stilsett

Stilsett, eller Cascaded Style Sheets som er det engelske navnet (forkortes CSS), er et viktig begrep for en som jobber med HTML 4.0. I spesifikasjonen for HTML 4.0 er alt av layout (dokumentets utseende) uønsket, og i stedet vil man at forfattere skal ta i bruk stilsett.

Fordelen med stilsett er at man skiller presentasjon (utseende) og dokumentets struktur. Dermed vil selve dokumentet være orientert mot god struktur, mens man i ettertid kan legge inn et stilsett som gir dokumentet et godt utseende. CSS har et utall av muligheter for blant annet å manipulere farger, skrift-type, skrift-størrelse eller elementers plassering.

Resultatet av en fokusering på god struktur vil være at en nettleser som ikke støtter stilsett allikevel vil kunne presentere informasjonen i dokumentet på en god måte. Det betyr at brukeren vil se et dokument som er oversiktlig og lettlest.

Ulempen med stilsett er at det krever en nettleser som støtter teknologien for at resultatet skal være godt. Dersom nettleser ikke støtter stilsett godt nok, vil brukeren kunne oppleve at dokumentet ikke er lesbart, og det er selvsagt ikke ønskelig. Andre resultater kan være at skrift-typen er feil, størrelsen er enten for liten eller for stor, eller at plasseringen av bilder er slik at teksten ikke kan leses.

Uansett er stilsett et skritt i riktig retning, fordi det fjerner presentasjons-informasjon fra struktur-informasjonen, og dermed letter arbeidet med å lage gode websider.

Dere skal nå få en liten smakebit på hva CSS kan gjøre. Vi skal vise hvordan vi ordner et vanlig problem med HTML: Hvordan får endret margene på teksten? En vanlig løsning er å bruke sitat-elementet, `<blockquote>` fordi de fleste nettlesere øker margene på begge sider for et sitat. Dette er også nevnt i HTML-standarden:

> *Visual user agents generally render BLOCKQUOTE as an indented block.*

Og senere omtaler den hvordan stilsett skal behandle `<blockquote>`:

> ***Note.*** *We recommend that style sheet implementations provide a mechanism for inserting quotation marks before and after a quotation delimited by BLOCKQUOTE in a manner appropriate to the current language context and the degree of nesting of quotations.*
>
> *However, as some authors have used BLOCKQUOTE merely as a mechanism to indent text, in order to preserve the intention of the authors, user agents should not insert quotation marks in the default style.*
>
> *The usage of BLOCKQUOTE to indent text is deprecated in favor of style sheets.*

Dette avsnittet omhandler nettlesere og deres mulighet for å sette inn sitat-tegn korrekt rundt sitater, og spesielt siste setning skal vi legge merke til. I og for seg burde det være allmennkunnskap at et sitat-element ikke skal brukes for å rykke inn tekst, men det har altså vært nødvendig å ta det med i standarden.

Hovedargumentet mot å bruke `<blockquote>` er fordi avsnittet blir semantisk feil. Avsnittet blir markert som et sitat, mens det i praksis brukes til å rykke inn tekst. Dette ser ikke brukere forskjell på såfremt deres nettleser ikke merker sitater spesielt, eller dersom nettleser har tale-syntese eller leselist. Med en gang sitatet

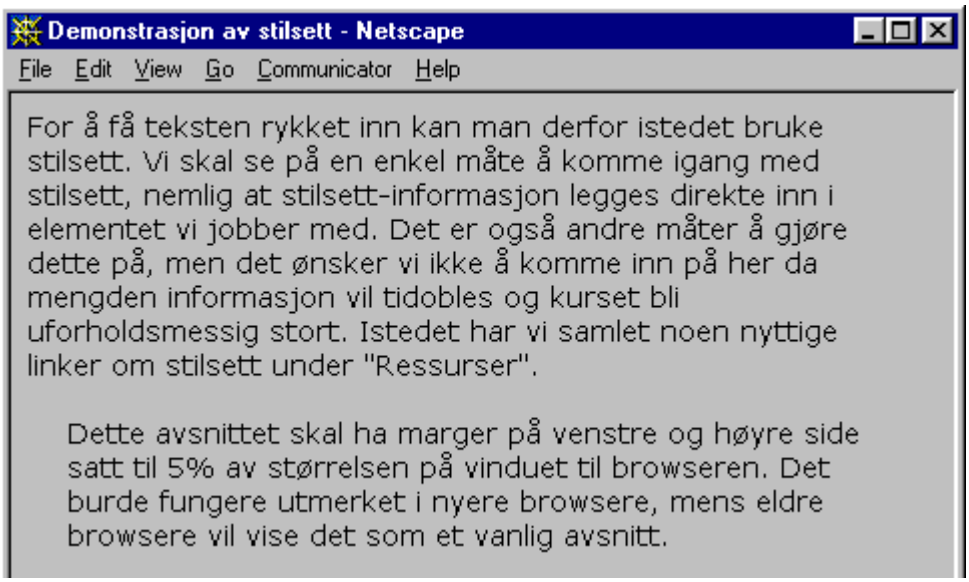
merkes spesielt vil brukeren ikke lenger oppfatte avsnittet som innrykket, men som et sitat.

For å få teksten rykket inn kan man derfor i stedet bruke stilsett. Vi skal se på en enkel måte å komme igang med stilsett, nemlig at stilsett-informasjon legges direkte inn i elementet vi jobber med. Det er også andre måter å gjøre dette på, men det ønsker vi ikke å komme inn på her da mengden informasjon vil tidobles og kapitlet bli uforholdsmessig stort.

Stilsett gir oss mulighet til å sette marger rundt et avsnitt ved å bruke `margin-left` og `margin-right`. Med disse settes henholdsvis venstre og høyre marg i avsnittet. Vi benytter oss av attributtet `style=""` og setter inn margene. Legg her merke til at stilsett-informasjonen skiller mellom attributt og parameter med et kolon ':', og at et semikolon ';' avslutter attributtet.

```
<p STYLE="margin-left: 5%; margin-right: 5%;">Dette avsnittet skal ha
marger på venstre og høyre side satt til 5 % av størrelsen på vinduet
til nettleser.  Det burde fungere utmerket i nyere nettlesere, mens
eldre nettlesere vil vise det som et vanlig avsnitt.</P>
```

For å demonstrere dette legger vi inn et avsnitt over som ikke har margene endret, slik at de er normale. Når vi så ser på dette i nettleser, kan det se slik ut:



Figur 7.1: Eksempel på avsnitt med økte marger ved hjelp av stilsett.

Som nevnt har stilsett et utall av muligheter. Dessverre kan ikke dette kapitlet ta for seg stilsett nærmere, ettersom det ville kreve en eller to kapitteler alene. Vi har samlet noen nyttige lenker under "Ressurser", og oppfordrer de som ønsker å lære mer om stilsett å bruke disse.

8- Skjema og JavaScript

8.1 Mål

Etter å ha gjennomgått denne studieenheten, skal du

- ha fått en innføring i bruken av skjema (forms) på web-sider
- vite hvorledes man bruker CGI til å behandle sider med skjema
- ha fått en innføring i JavaScript og hvordan dette kan utnyttes på web

8.2 Skjema

Et skjema på en web-side kan gi deg mulighet til å gjøre spørreundersøkelser, la brukerne sende inn ønske om å få tilsendt katalogmateriell, og så videre. Det brukeren taster inn eller velger blir sendt videre til et program som behandler dataene, og resultatet kan enten vises til bruker på skjermen, lagres i fil, eller for eksempel sendes i e-post til noen. Ved hjelp av skjema og bakenforliggende programmer lar man dermed muligheten til å gjøre all den form for behandling av data man kunne ønske.

Det er dog viktig å huske på at man er nødt til å ha et program i bakgrunnen. HTML er ikke et programmeringsspråk og har derfor ikke mekanismer for behandling av data. Det finnes flere måter å løse problemet med behandlingen av data på, men en av de vanligste er å benytte seg av CGI.

8.3 CGI

CGI står for Common Gateway Interface. Ved hjelp av CGI får man et felles grensesnitt mellom web-serveren og programmer som serveren kjører på oppfordring av brukere. Det vil altså si, brukeren følger en lenke som i virkeligheten er et program som kjøres på serveren. Den web-siden brukeren så får på skjermen er utskrift av programmet som ble kjørt. Hva programmet gjør er opptil forfatteren av web-siden, brukeren lar ingen mulighet til å omprogrammere eller på annen måte påvirke hva programmet gjør.

Fordelen med CGI er at du kan skrive programmene i det språket du ønsker så lenge det er mulig å la programmet ta input fra serveren og dytte output tilbake. Under windows-plattformen er det vanlig å skrive i Visual Basic eller C++, mens på UNIX-plattformer er det stort sett Perl som råder. Grunnen til det sistnevnte er at Perl er meget godt egnet til behandling av tekst, hvilket er det man gjør med CGI.

Ulempene med CGI er at det bruker store mengder maskinkraft, samt at det medfører en sikkerhetsrisiko. Serveren må ha mulighet til å kjøre flere instå nser av programmet samtidig, hvilket spiser maskinkraft. En av de vanligste årsakene til at servere får pusteproblemer er at programmene som kjøres via CGI tar opp all maskinkraft som er ledig. I tillegg vil bruken av CGI medføre en større mulighet for brudd på sikkerheten ettersom det åpner en ny tjeneste.

Dette kurset tar for seg HTML og vil derfor ikke omhandle selve programmeringen. De som ønsker det kan i så fall sette seg inn i det programmeringsspråket de ønsker å bruke, samt skaffe seg tilgang til en server hvor de kan prøve ut programmene de skriver.

I dette kapitlet vil vi gp igjennom en rekke skjema-eksempler, og for å vise hvordan de virker vil vi bruke ASP-skripter-som ligger på kursserverenxxxx. Det vil gå fram av eksemplene hvilken URL og program som brukes. Dette er gjort for å gi muligheten til å prøve ut skjemaer i praksis uten å ha programmeringskunnskaper.

8.4 Et enkelt skjema

Vi kan ta utgangspunkt i et enkelt skjema som lar brukeren fylle inn navn, adresse, telefonnummer og e-post-adresse og sender alt til deg via e-post. Dette skjemaet vil dermed ikke benytte seg av server-srcripting, men i stedet sende det brukeren taster inn direkte til deg. Etter at vi har laget skjemaet skal, vi se hvorledes vi kan gjøre det samme ved hjelp av CGI og et program.

8.4.1 Skjemaets oppbygging

Et skjema kan omtrent sees på som et selvstendig dokument i dokumentet. Dette fordi det er svært få begrensninger på hva skjemaet kan inneholde, avsnitt med tekst, tabeller, bilder, og så videre, går helt flint. Dette betyr at du står veldig fritt med tanke på hvorledes du ønsker å utforme skjemaet.

Et skjema er et blokk-nivå-element, start-taggen er <**form**> og slutt-taggen er **</form**>

For at skjemaet skal fungere trenger start-taggen en del hjelp. Ettersom det er umulig å få gjort noe med informasjonen i skjemaet uten hjelp fra eksterne applikasjoner (programmer), må vi fortelle nettleseren hvor den finner applikasjonen. Dette gjøres med attributtet **action**="". Attributtet skal inneholde en URL, hvilket betyr at den enten kan ha en URL som peker på et program vi ønsker å kjøre, eller den kan bruke "**mailto:**", som du kan lese om i kapittel 1, og sender innholdet av skjemaet til deg via e-post.

I tillegg til action = " " må vi hjelpe <form> med et attributt til. Dette attributtet skal fortelle noe om hvordan nettleseren skal behandle det brukeren taster inn for det sendes videre. Attributtet er **method**="", og den kan ta to verdier: **get** eller **post**. Det er viktig å vite forskjellen mellom de to, og vi vil ta i bruk minst ett eksempel med hver av metodene.

Før vi ser nærmere method="" skal vi i stedet ta en titt på en av de mange byggeklossene vi har til skjemaet vårt. Dette fordi det er vanskelig å forstå hvordan method="" fungerer uten å ha med seg en byggekloss til å lage eksempel med.

Den første byggeklossen vår skal la brukeren taste inn en linje med tekst. Dette brukes gjerne for inntasting av navn, e-post-adresser, telefonnumre, og lignende. Taggen man bruker til dette er **<input>. <input>** er en byggekloss vi kommer til å bruke mye, ettersom den har et attributt, **type=""** som lar oss spesifisere hvordan brukeren kan taste inn noe.

Vi ønsker å ha et tekstfelt og da bruker vi <**input type**="text">. I tillegg kan vi fortelle nettleseren hvor stort vi ønsker feitet skal være, det vil si antall tegn. I eksemplet vårt ønsker vi at brukeren skal skrive inn navnet sitt (både fornavn og

etternavn) og sende det til oss via e-post. Koden vi bruker ser da slik ut (vi har utelatt resten av dokumentet og konsentrer oss kun om innholdet i dokument-kroppen):

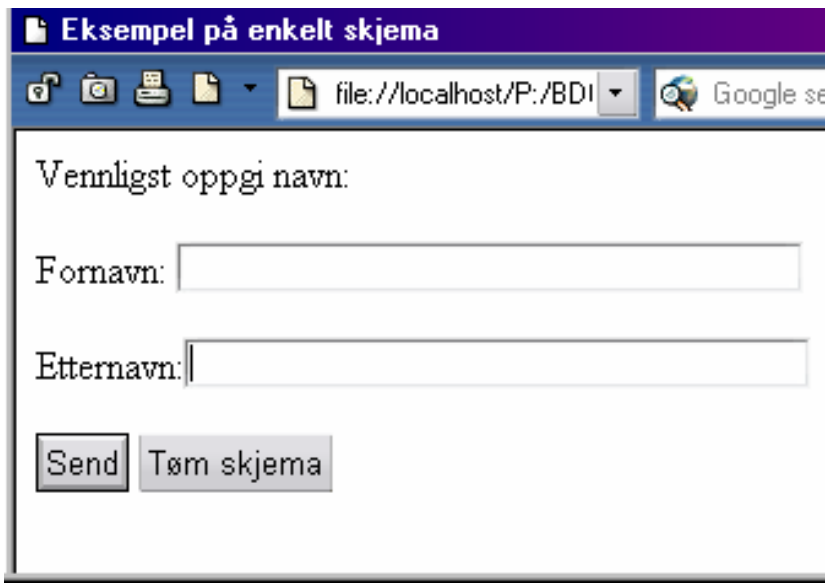
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
   <head>
    <title>Eksempel på enkelt skjema </title>
   </head>
   <body>
    <form action="mailto:jarleha@stud.hisf.no" method="post">
     <p>Vennligst oppgi navn:</p>
     <p>Fornavn: <input type="text" name="Fornavn" size="40"> </p>
     <p>Etternavn:<input type="text" name="Fornavn" size="40"> </p>
     <p><input type="submit" value="Send"> <input type="reset" value="Tøm
skjema">
    </form>
   </body>
</html> <input type=”text” size=”40” name=”Navn”></p>
```

skript0801.html: Eksempel på et enkelt skjema.

Her er det brukt to elementer i tilleg til text-elementet: <input type=”submit”> og <INPUT TYPE=”reset”>. Disse to elementene lar brukene henholdsvis sende avgårde innholdet i skjemaet, eller slette alle felter og begynne på nytt. Som vi senere skal se, vil det siste også medføre at knapper som er huket av vil bli satt tilbake til standard-verdiene de hadde når web-siden med skjemaet ble lastet inn for første gang. Husk også å sette inn din egen e-post-adresse istedenfor “min-adresse@stud.hisf.no” slik at posten faktisk kommer fram.

Legg også merke til bruken av attributtet name=”” for å gi tekstfeltet et navn vi senere kan bruke til å identifisere innholdet i feltet. Vi har også brukt size=” 40”, som forteller nettleseren hvor mange tegn vi ønsker at feltet skal ha. Det er mulig å taste inn flere tegn i feitet, mens size=”” spesifiserer hvor mange tegn feltet skal ha på skjermen. Vanlige redigeringsmuligheter, som å gå tilbake ett tegn, til begynnelsen av linjen, og så videre, pleier å være tilgiengelig i netllesere når man taster inn verdier i felter som dette.

I tillegg har det blitt lagt inn litt fylltekst, samt at de forskjellige elementene er lagt inn i avsnitt slik at skjemaet er godt strukturert også. når du legger dette inn i en web-side og tar en titt på det i nettleseren din, kan det se slik ut:



Figur 8.1: Eksempel på et enkelt skjema

Som du ser har vi fått et tekstfelt hvor brukeren kan taste inn navnet sitt.

Dersom han eller hun klikker Send-knappen vil innholdet av det brukeren tastet inn bli sendt til deg via e-post.

Skriv inn koden til skript0801.html, arkiver det i en mappe08 og se hvordan det blir seende ut i nettleseren.

8.5 De to metodene - get og post

Som nevnt tar method="" to verdier: "get" eller "post". Nå som vi har latt brukeren taste inn noe, kan vi se på forskjellen mellom de to verdiene.

Du har husket å prøve ut eksemplet, og sett på hva resultatet er? Du la kanskje merke til at posten ble sendt til deg, mens innholdet var et vedlegg som sa eksempelvis "NameOla+Nordmann"? Det er slik method="post" fungerer. Nettleseren går til URL'en som er oppgitt i action="" og gir serveren et vedlegg som inneholder det som brukeren har tastet inn.

Dersom du bruker method="get", vil nettleseren i stedet ta URL'en du oppgir i action="", legge til verdiene som er tastet inn separert med et spørsmålstegn, og så be om den nye URL'en fra serveren. Vi kan tenke oss at vi har URL'en "http://stud-vevtjener/skjematest.asp" i action="" og brukeren taster inn "Ola Nordmann" i navnefeltet vårt. Nettleseren vil da ta URL'en vi hadde, legge til et spørsmålstegn, og så legge til verdien som ble tastet inn. Den nye URL'en blir da "http://stud-vevtjener/skjematest.asp?Navn=Ola+Nordmann".

De to metodene har hvert sitt bruksområde. Spesifikasjonene for HTML 4.0 tilsier at dersom skjemaet har bivirkninger, det vil si for eksempel endrer innholdet i en database eller abonnement til en tjeneste, skal method="post" brukes. Dersom skjemaet ikke har bivirkninger, for eksempel dersom man bruker det for å gjøre søk på et web-sted, skal method= "get" brukes.

Du kan prøve ut begge metodene med eksemplet vårt, og legge merke til hvorledes den ene metoden virker, mens den andre metoden ikke gjør det. Prøv også å finne ut hvorfor den ene metoden ikke virker, og hva som skulle vært nødvendig for at den skulle fungere.

8.6 Flere byggeklosser

Vi skal nå ta for oss de andre typene av byggeklosser vi har. La oss starte med en variant av tekstfeltet vi hadde: passord-felt.

8.6.1 Passord-felt

Et passord-felt er likt et tekst-felt, men brukeren vil ikke se i klartekst det han/hun taster inn. Dette er selvfølgelig fordi feltet er ment å brukes til passord, og man ønsker ikke at personer skal stå bak en og lese passordet man taster inn. Browsere er derfor oppfordret til å sørge for at hvert tegn som er inntastet ikke er synlig i klartekst, ved for eksempel å vise hvert tegn som et asterisk.

Vi kan nå utvide skjemaet vårt og la det etterligne en innloggingsskjerm, hvor brukeren taster inn brukernavn og passord og deretter velger "Logg inn". På den måten kan man ved hjelp av CGI skape et delvis sikkert web-sted.

Vi endrer skjemaet vårt og lar det nå se slik ut:

```
<body>
    <form action="http://stud-vevtjener/kursform.asp" method="post">
    <p>Pålogging</p>
    <p>Brukernavn: <input type="text" name="Brukernavn" size="10"> </p>
      <p>Passord:<input type="password" name="Passord" size="20"> </p>
    <p><input type="submit" value="Logg inn">
       <input type="reset" value="Slett felter">
    </p>
   </form>
</body>
```

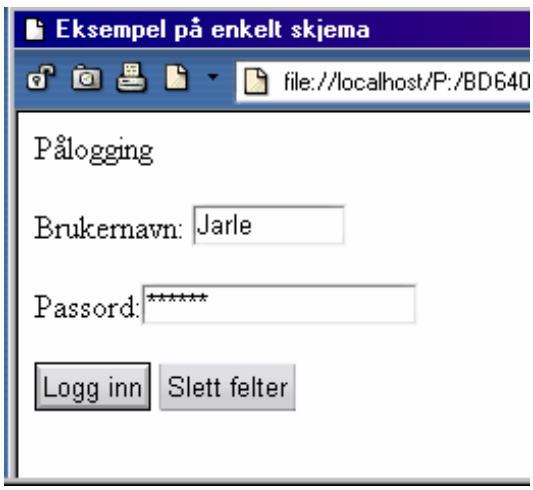
skript0802.html: Eksempel på bruk av passord-felt.

Legg her merke til at vi bruker method=”post”, ettersom vi ønsker at nettleseren skal sende det inntastede brukernavnet og passordet som et vedlegg. Dersom vi hadde brukt method=”get” ville den resulterende URL’en vist passordet i klartekst. URL’en i action=”” ville normalt pekt på et program som tar seg av å sjekke om brukernavn og passord stemmer. I vårt eksempel har vi i stedet brukt et program på web-server som heter “kursform.pl”. Deffe programmet får tilsendt verdier med et skjema og skriver så de tilsendte verdiene ut på en ny web-side som brukeren kan se. Dette for at vi lett kan prøve ut skjemaene vi lager.

Det er verd å merke seg at et slikt sikkerhetssystem ikke er noe man burde stole så altfor mye på . Dersom sikkerheten er viktig, burde man istedet se på andre løsninger med kryptert overføring, i vårt eksempel vil passordet bli overført i klartekst til ASP-scriptet.

Som det går frem av eksemplet er forskjellen mellom et vanlig tekst-felt og et passord-felt at vi bruker type=”password” istedenfor type=”text”. Det er ikke noen andre forskjeller mellom de to feltene.

I eksemplet har vi også satt verdier for submit- og reset-knappene. På den måten får vi da den teksten som vi ønsker, og dermed et mer forståelig brukergrensesnitt. I vårt tilfelle gjør den ene knappen at brukernavn og passord sendes videre, det vil si at brukeren logger inn på systemet. Derfor har vi merket submit-knappen med “Logg inn” ved hjelp av attributtet value=”Logg inn”. På samme måte setter vi også teksten til reset-knappen med value=”Slett felter”. når du legger eksempelet inn på en web-side, kan det se slik ut:



Figur 8.2: Skjema-eksempel med pålogging

Legg her merke til hvor skjevt de to feltene står i forhold til hverandre.

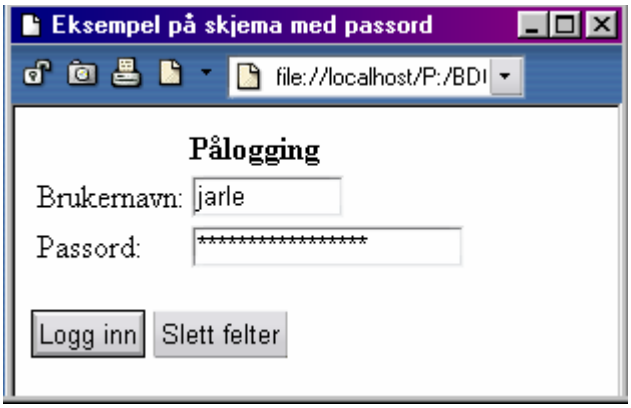
Vi bør få til at de to feltene brukeren taster inn verdier i står rett under hverandre. Det gjøres best ved å benytte muligheten til å la et skjema inkludere det vi ønsker, vi lager en tabell!

Eksemplet vårt blir da slik:

```
<body>
 <form action="http://stud-vevtjener/kursform.asp" method="post">

  <table>
   <tr><th colspan="2">Pålogging</th></tr>
   <tr><td>Brukernavn:</td><td><input type="text" name="Brukernavn" size="10"></td></tr>

   <tr><td>Passord:</td><td><input type="password" name="Passord" size="20"> </td></tr>
  </table>

<p><input type="submit" value="Logg inn">
     <input type="reset" value="Slett felter">
  </p>
 </form>
</body>
```

skript0802b.html: Bruk av tabell for layout i skjemaer.

Og skjemaet, når det settes inn på en web-side, kan da bli seende slik ut:



Figur 8.2: Skjemaeksempel med pålogging. Tabell for justering av layout.

8.7 Valg

Ofte vil man at brukeren skal kunne velge fra alternativer man gir dem. Skjema på web-sider har to muligheter for slike valg. Vi skal først se på den vanlige avkrysningsboksen.

8.7.1 Avkrysning

Det kan være ønskelig å gi brukeren mulighet til å krysse av for ett eller flere alternativer som vi gir dem. Det kan for eksempel være at de ønsker om å få tilsendt flere katalogtyper eller annet informasjonsmateriell. Dette er avkrysnings-bokser godt egnet til. Egenskapene til boksene er at brukeren kan krysse av en eller flere av boksene. Programmet som behandler det brukeren taster inn bør sjekke etter om brukeren faktisk har krysset av.

I dette eksemplet skal vi lage et skjema hvor brukeren skriver inn fomavn, etternavn, adresse og postnummer. I tillegg skal han/hun kunne krysse av i en eller flere av fire bokser som sier hvilke kataloger personen ønsker tilsendt. Vi bruker da attributtet **type**=”checkbox”. Igjen bruker vi en tabell for å gi skjemaet ett litt bedre utseende, og firmaet vi lager skjemaet for lager bedriftstelefonkataloger for Norge.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
   <head>
   <meta name="author" content="JARLE.HAVIK@HISF.NO">
      <title>Eksempel på skjema med avkrysningabokser </title>
 </head>
 <body align="center">
  <h2>Katalogservice L/L </h2>
  <br>
  <p>Tast inn navn og adresse, samt kryss av for telefonkatalogene du ønsker tilsendt.
     Katalogene vil være deg i hende så fort som mulig.
  </p>
  <form action="http://stud-vevtjener/kursform.asp" method="post">
  <table>
    <tr><th colspan="2" align="left">Bestillingsskjema:</th></tr>
    <tr><td>Fornavn:</td><td><input type="text" size="30" name="fornavn"></td></tr>
    <tr><td>Etternavn:</td><td><input type="text" size="30" name="etternavn"></td></tr>
    <tr><td>Adresse:</td><td><input type="text" size="20" name="adresse"> </td></tr>
    <tr><td>Postnr:</td><td><input type="text" size="20" name="postnr" > </td></tr>
  </table>
  <p><input type="checkbox" name="oest">Telefonkatalogen for Østlandet<br>
     <input type="checkbox" name="vest">Telefonkatalogen for Vestlandet<br>
     <input type="checkbox" name="soer">Telefonkatalogen for Sørlandet<br>
     <input type="checkbox" name="nord">Telefonkatalogen for Nordlandet<br>
  </p>
  <p><input type="submit" value="Send skjema">
     <input type="reset" value="Tøm skjema">
  </p>
 </form>
 </body>
</html>
```

skript0803.html: Skjemaer med avkrysningsbokser.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript00803.html`.

Resultatet i nett-leseren kan se slik ut

Figur 8.3: Skjema med avkrysnings-bokser.

Det er igjen enkelte ting man bør legge merke til. Første punkt er kanskje at post-sted ikke er tatt med som tekst-felt. Ettersom det er mulig å skaffe en liste over norske poststeder koblet til post-nummer, og et program enkelt kan søke gjennom en slik liste, er det å skrive inn poststedet overflødig. En taster i stedet kun inn nummeret, og lar datamaskinen ta seg av resten.

Videre har vi fire avkrysningsbokser brukerne kan benytte. Hver av boksene har sitt navn, ettersom de ikke er avhengige av hverandre.

8.7.2 Avhengige avkrysningsbokser

I andre sammenhenger kan det være nyttig å ha avkrysningsbokser som er gjennsidig utelukkende. Det vil si, velger man en ting velger man automatisk ikke en annen ting. Typisk eksempel er ja/nei-spørsmål, valg av kjønn i spørreundersøkelser, og lignende.

Vi kan ta det forrige eksemplet vi hadde og endre litt på det. Denne gangen bruker vi det til å legge inn litt personalia i et register. Et av feltene ønsker vi skal være kjønn, og derfor bruker vi to bokser til dette

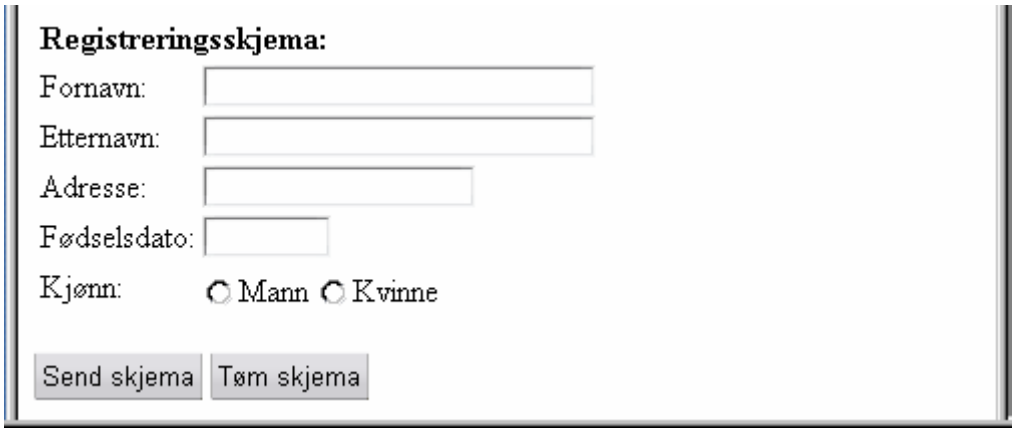
```
<form action="http://stud-vevtjener/kursform.asp" method="post">
   <table>
     <tr><th colspan="2" align="left">Bestillingsskjema:</th></tr>
     <tr><td>Fornavn:</td><td><input type="text" size="30" name="fornavn"></td></tr>
     <tr><td>Etternavn:</td><td><input type="text" size="30" name="etternavn"> </td></tr>
     <tr><td>Adresse:</td><td><input type="text" size="20" name="adresse"></td></tr>
     <tr><td>Fødselsdato:</td><td><input type="text" size="8" name="fdato"></td></tr>
     <tr><td>Kjønn:</td> <td><input type="radio"name="gender" value="mann">Mann
                           <input type="radio"name="gender" value="kvinne">Kvinne
                       </td>
     </tr>
   </table>
   <p><input type="submit" value="Send skjema">
      <input type="reset" value="Tøm skjema">
   </p>
</form>
```

skript0803b.html: Skjemaer med radioknapper.

Skjemaet introduserer den nye typen knapper. Disse bruker attributtet type=”radio”, og kalles da også for radio-knapper. Legg merke til at begge knappene har samme navn. Hadde de ikke hatt samme navn ville ikke nettleseren skjønt at de hører sammen, og dermed ville man ikke fått avhengigheten.

Videre har disse knappene også fått en verdi ved hjelp av value="". Når man bruker radio-knapper, er man nødt til å gi hver knapp en verdi, slik at nettleseren kan sende riktige data videre til serveren. Begge knappene deler samme navn, hvilket betyr at dersom de ikke får noen verdi vil nettleseren sende eksempelvis "navn=on" uansett hvilket valg som brukes. Gir man knappene verdier slik vi har gjort, sendes “sex=mann” istedenfor “sex=on”.

Satt inn på en web-side, kan skjemaet se slik ut:



Figur 8.4: Skjema med radio-knapper.

8.7.3 Sette knapper valgt

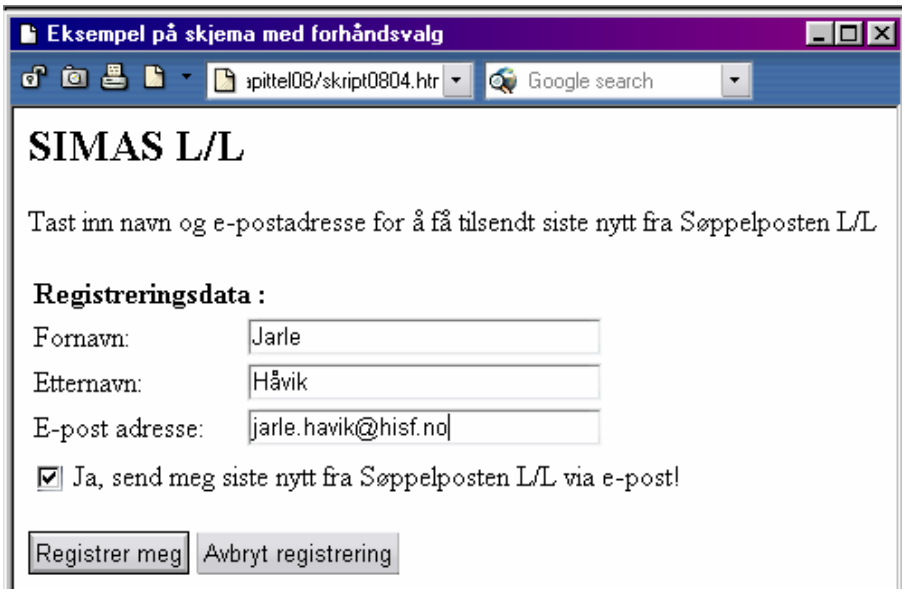
I enkelte skjemaer kan det være ønskelig å sette en avkrysningsboks eller en radio-knapp som valgt (standardverdi). Det vil si at når brukeren får opp siden for første gang, er en eller flere av knappene allerede huket av. Brukeren kan da selvfølgelig fjerne avkrysningen, eller velge andre alternativer. Ved trykk på en reset-knapp vil standardverdiene igjen lastes inn, og de knappene som da er satt som valgt vil igjen bli valgt.

Et vanlig eksempel på dette er registreringsskjema hos programvareleverandører eller nettsteder. Disse vil gjerne sende deg e-post med nyheter om ting som skjer, og da har de en avkrysningsboks merket for eksempel “Jeg vil gjerne motta nyheter via e-post.” Vi kan lage et eksempel på et slikt skjema.

```
<form action="http://stud-vevtjener/kursform.asp" method="post">
  <table>
   <tr><th colspan="2" align="left">Registreringsdata :</th></tr>
   <tr><td>Fornavn:</td><td><input type="text" size="30" name="fornavn"></td></tr>
   <tr><td>Etternavn:</td><td><input type="text" size="30" name="etternavn"> </td></tr>
   <tr><td>E-post adresse:</td><input type="text" size="30" name="adresse"> </td></tr>
    <td><input type="checkbox"name="nyheter" value="ja" checked>
      Ja, send meg siste nytt fra Søppelposten L/L via e-post!
    </td>
   </tr>
  </table>
  <p><input type="submit" value="Registrer meg">
     <input type="reset" value="Avbryt registrering">
  </p>
</form>
```

skript0804.html: Skjemaer med forhåndsvalg.

Dette skjemaet har også introdusert et nytt attributt, **checked**. Denne settes inn i <input> når du har enten type="checkbox" eller type="radio" og det aktuelle valget som standardverdi. Du kan selvsagt ikke sette mer et valg som standardverdi for en radio-knapp!. Dette går også frem av figur 46. når vi så legger dette skjemaet inn i en web-side og ser på det i nettleseren, vil resultatet kunne se slik ut:



Figur 8.5: Skjema med avkrysnings-bokser.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0804.html`.

Prøv deg frem med dette eksemplet.

Sjekk at du kan fjerne avkrysningen i boksen, trykke på resetknappen og dermed få avkrysningen tilbake.

8.7.4 Skjulte felt

Den siste typen felt vi skal se på er skjulte felt. Disse bruker man til å legge inn data i skjemaet som brukeren ikke ser. Dette kan være en e-postadresse dataene skal sendes til, eller det kan brukes til å ta med seg data mellom flere web-sider. Det siste er en egenskap som ikke finnes i HTML, men som man ved hjelp av et CGI-program kan få til. Programmet lar da ett eller flere skjulte felt som inneholder verdier man vil ta med seg videre i beregningene, for eksempel en unik identifikasjonsverdi som hver bruker får.

Vi kan vise et eksempel, hvor vi bruker et skjult felt til nettopp å definere en e-post-adresse som inntastede verdier skal sendes til. Scriptet vi bruker er "xxxx.asp" som sender inntastede data til en angitt e-post-adresse. Skjemaet lar brukeren skrive inn navn, e-post-adresse og URL til oppgaveløsningen og sende det med e-post til brukeren.

```
<form action="http://stud-vevtjener/kursform.asp" method="post">

    <input type="hidden" name="tiladr" value="min.epost@stud.hisf.no">

    <table>
      <tr><th colspan="2">Registreringsdata :</th></tr>
      <tr><td>Fornavn:</td><td><input type="text" name="fornavn"></td></tr>
      <tr><td>Etternavn:</td><td><input type="text" name="etternavn"></td></tr>
      <tr><td>E-post adresse:</td><td><input type="text" name="adresse"></td></tr>
      <tr><td>URL:</td><td><input type="text" name="url"></td></tr>
    </table>
    <p><input type="submit" value="Send inn">
       <input type="reset" value="Tøm felter">
   </p>
</form>
```

skript0805.html: Skjemaer med skjult felt.

Skriv inn koden over i kroppen og arkiver den med filnavnet `skript0805.html`. Husk på å endre adressen i <input type="hidden" name="tiladr" value="min.epost@stud.hisf.no"> til din egen gyldige e-post-adresse. Sjekk så resultatet i din nettleser:

Figur 8.6: Skjema med skjult felt

Legg merke til hvordan det skjulte feltet ikke kan sees på skjemaet i det hele tatt. Feltet forteller scriptet hvilken e-post-adresse informasjonen skal sendes til, og det fungerer i praksis, men brukeren har ingen mulighet til å på virke dette feltet ettersom han/hun ikke ser det uten å titte på HTML-koden.

8.7.5 Menyer

Vi har nå vært gjennom de forskjellige mulighetene vi har med <input>, men det er ikke alle muligheten vi har med skjemaer. Nå skal vi se på enda en mulighet for å gi brukeren muligheter til å velge noe. Tidligere brukte vi avkrysningsbokser og radio-knapper, denne gangen er det snakk om en liste.

Elementet vi bruker er <**select**>. I motsetning til <input> som bare har en start-tagg, har <select> også en slutt-tagg </select>. Select gir oss en nedtrekksmeny, hvor brukeren kan velge en eller flere alternativer. Dette er mer oversiktlig når det er snakk om et stort antall muligheter hvor vanlige avkrysningsbokser ville skapt en lang liste.

<select> kan sees på som en liste. Det vil si at den trenger listepunkter for å fungere. Det er derimot ikke listepunkter (<li>) den skal inneholde, men valg (opsjoner). Disse defineres med taggen <**option**>. Hvert valg har sin egen <option> og kommer opp på listen brukeren kan velge fra. Du kan, dersom du ønsker, definere verdien til valget med value="", men det er valgfritt. Dersom du ikke spesifiserer en verdi, vil innholdet av valget, det vil si teksten som kommer etter den avsluttende >, være verdien.

Vi kan lage et eksempel hvor skjemaet skal registrere nye brukere på en online-tjeneste. Denne online-tjenesten er hyggelig mot sine nye brukere og sender dem en gratis t-skjorte. For ordens skyld har de t-skjorter i alle regnbuens farger, og dette gis så som valg til brukerne. La oss se hvordan dette kan kodes

```
<form action="http://stud-vevtjener/kursform.asp" method="post">
   <input type="hidden" name="tiladr" value="jarleha@stud.hisf.no">
   <table>
     <tr><th colspan="2">Registreringsskjema :</th></tr>
     <tr><td>Fornavn:</td><td><input type="text" name="fornavn"></td></tr>
     <tr><td>Etternavn:</td><td><input type="text" name="etternavn"></td></tr>
     <tr><td>Adresse:</td><td><input type="text" name="adresse"></td></tr>
     <tr><td>Postnr:</td><td><input type="text" size="20" name="postnr"> </td></tr>
   </table>

   <p>Ønsker t-skjorte:<input type="radio"name="skjorte" value="ja">Ja
                       <input type="radio"name="skjorte" value="nei">Nei<br>


    Farge: <select name="farge"> <option>Hvit<option>Lysegrå
             <option>Mørkegrå <option>Lyseblå <option>Marineblå
             <option>Lysegrønn <option>Mørkegrønn<option>Lilla <option>Vinrød
             <option>Signalrød <option>Oransj <option>Gul <option>Sort
           </select>
   </p>

   <p><input type="submit" value="Send inn">
      <input type="reset" value="Tøm felter">
   </p>

</form>
```

skript0806.html: Skjema med liste.

Listen over mulige fargevalg er ganske lang, og vi har derfor brukt <select>, slik at listen ikke tar uforholdsmessig stor plass på web-siden. Attributtet name="" brukes på samme måten som ved <input>, og valget blir sendt videre til CGI-scriptet med eksempelvis "farge=hvit", slik valg tidligere også har blitt sendt.

Ettersom du ikke har mulighet til å skreddersy scriptene selv, vil du kunne få listet opp farge-valget på utskriften fra http://stud-vevtjener/kursform.asp selv om du velger "Nei" på spørsmålet "ønsker t-skjorte?" I praksis ville man selvfølgelig laget et eget script som fjernet fargevalget fra utskriften, slik at den personen eller det programmet som behandlet registreringen ikke fikk unødvendige data tilsend.

Når dette så legges inn på en web-side, kan det se slik ut (viser her nettleseren med nedtrekks-menyen oppe, slik brukeren vil se det):

Figur 8.7: Eksempel på web-side med nedtrekksmeny.

Prøv dette ut i praksis med xxxx.asp og se hvordan det fungerer.

Størrelse, flervalg og forutvalgte felt

Du la kanskje merke til at valget bare ble vist med 1 linje på skjermen. Dette, samt muligheten for å la brukere velge flere valg, skal vi se på nå.

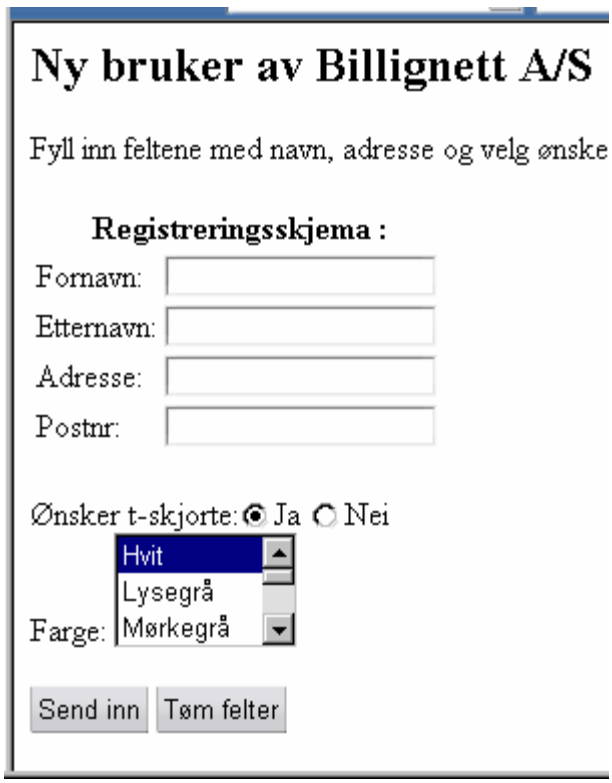
Størrelsen på denne valg-menyen bestemmes av attributtet size="" som vi setter sammen med <select>. Parameteret vi setter inn er et tall som forteller hvor mange linjer som skal vises på skjermen. Menyen går da gjerne over fra å være en nedtrekksmeny, slik vi så i forrige eksempel, til å bli en rullefelt-meny. Vi kan legge til size="" i det forrige eksemplet og se på resultatet. I tilleg kan vi sette Ja som standardverdi på radio-knappen om ønskes T-skjorte:

```
<p>Ønsker t-skjorte:<input type="radio"name="skjorte" value="ja" checked>Ja
                    <input type="radio"name="skjorte" value="nei">Nei<br>
     Farge: <select name="farge" size="3"> <option>Hvit<option>Lysegrå
                <option>Mørkegrå <option>Lyseblå <option>Marineblå
                <option>Lysegrønn <option>Mørkegrønn<option>Lilla<option>Vinrød
                <option>Signalrød <option>Oransj <option>Gul <option>Sort
            </select>
</p>
```

skript0806b.html: Skjema med liste hvor flere valg vise.

Skriv inn koden og arkiver denne som skript0806b.html. I nettleseren vil siden kunne se ut som dette:

Figur 8.8.~ Eksempel på rullefeltsmeny med størrelse satt.



Som vi ser går vi her over fra å ha en nedtrekksmeny hvor brukeren valgte ned-pil for å få opp en full liste, til å ha en rullefeltsmeny med 3 linjer vist av gangen, og brukeren kan velge ett av valgene.

Begge eksemplene så langt gir brukeren kun mulighet til å velge 1 verdi. Ved valg av for eksempel produkter, kan det være praktisk at brukeren får muligheten til å velge flere verdier. Dette er også mulig, vi bruker da attributtet **multiple**. Dette attributtet forteller nettleseren at den skal gi brukeren mulighet til å velge mer enn en verdi. Det er her verdt å merke seg at nettleseren vil bruke samme navnet på alle verdiene, hentet fra **<select name="">**, og sende hvert valg for seg. Det vil si, velger brukeren tre valg vil nettleseren sende tre verdier, alle med samme navn.

La oss se på et eksempel med fylkesinndelte telefonkataloger:

```
<form action="http://stud-vevtjener/kursform.asp" method="post">
  <table>
  <tr><th colspan="2" align="left">Bestillingsskjema:</th></tr>
  <tr><td>Fornavn:</td><td><input type="text" size="30" name="fornavn"></td></tr>
   <tr><td>Etternavn:</td><td><input type="text" size="30" name="etternavn"></td></tr>
  <tr><td>Adresse:</td><td><input type="text" size="20" name="adresse"> </td></tr>
  <tr><td>Postnr:</td><td><input type="text" size="20" name="postnr" > </td></tr>
  </table>

  <p>Jeg ønsker å få tilsendt følgende telefonkatalog(er) :<br>
    (Du kan merke flere enn 1 dersom du trenger kataloger for flere fylker)<br>
   <select name="katalog" multiple size="5"> <option>Oslo<option>Akershus
       <option>Østfold<option>Vestfold<option>Aust-Agder<option>Vest-Agder
       <option>Rogaland <option>Telemark<option>Buskerud <option>Hordaland
       <option>Hedmark<option>Oppland<option>Sogn og Fjordane
       <option>Møre og Romsdal <option>Sør-Trøndelag <option>Nord-Trøndelag
       <option>Nordland <option>Troms <option>Finnmark
   </select>
  </p>

  <p><input type="submit" value="Send skjema">
     <input type="reset" value="Tøm skjema">
  </p>
</form>
```

skript0807.html: Skjema med flervalgsliste.

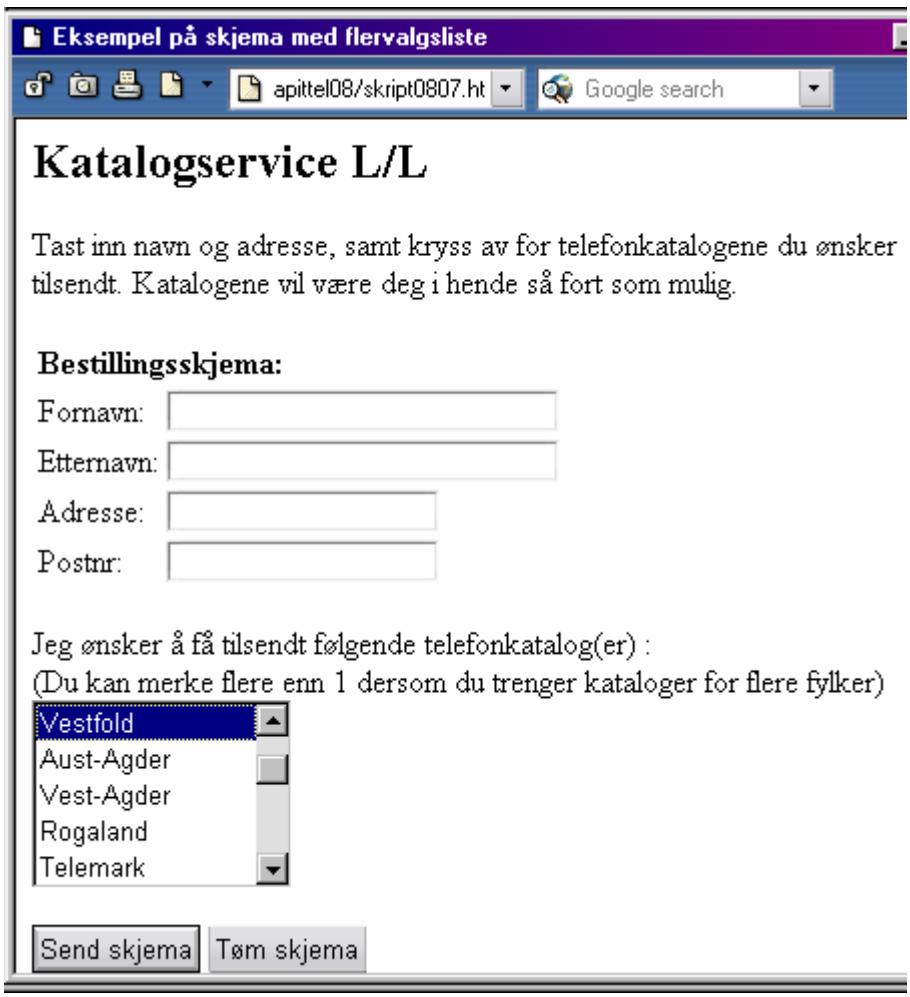
Her det igjen brukt en rullefelt-meny, og størrelsen er satt til 5, slik at den ikke tar for mye plass på web-siden.

Skriv inn koden og arkiver denne som skript0807.html

Prøv også å se hva som skjer dersom du fjerner size="5" fra koden.

Sjekk også hvilke verdier som blir sendt til scriptet når brukeren velger 1 eller flere kataloger.

Siden vil kunne se slik ut:



Figur 8.9: Eksempel på meny med flervalg

8.7.6 Flere linjert tekstfelt - `textarea`

Vi har sett hvordan vi kan bruke <input type=”text”> for å gi brukeren tilgang til å skrive inn tekst. Dette tekstfeltet er begrenset til en linje, og egner seg derfor kun til for eksempel navn, adresser, postnummer, telefonnummer og lignende. Ved for eksempel brukerunder-søkelser kan det være praktisk å gi brukerne tilgang til å skrive inn flere linjer med tekst, og redigere dette med vanlige redigerings-muligheter.

For å gjøre dette mulig bruker vi et nytt element: <**textarea**>.Det er ikke lenger snakk om et enkelt tekst-felt, men et område med tekst. <textarea> har både start- og slutt-tagg. Slutt-taggen er <**/textarea**>.<textarea> har 3 attributter.

Det første er `name=""` som virker slik vi tidligere har brukt den, ved at den gir feltet ett navn som senere brukes når man sender data fra nettleseren til et script. De to andre kontrollerer bredde og høyde på tekst-området og er `rows=""` og `cols=""`. Bredde og høyde er her da oppgitt i antall rekker i høyden og antall tegn i bredden (kolonner).

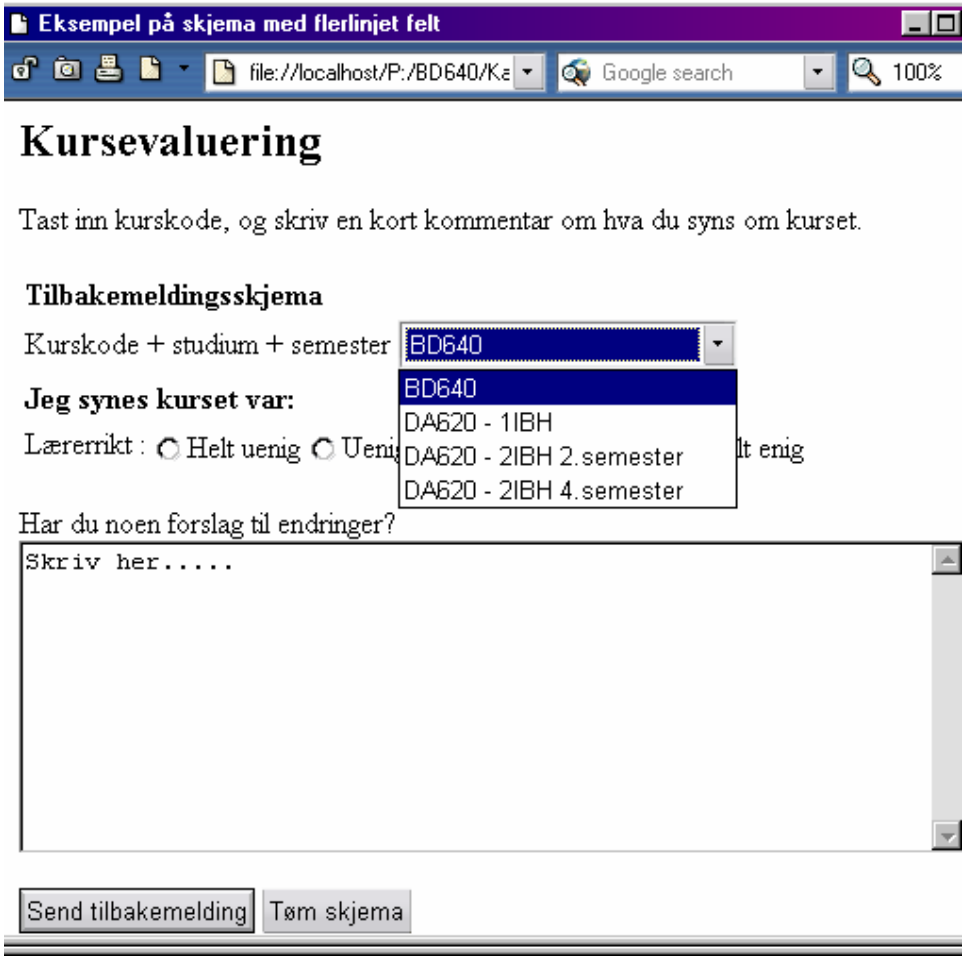
Vi kan lage et enkelt eksempel som gir der som studenter mulighet til å gi meg en tilbakemelding om hva dere synes om dette kurset.

```
<body>
 <h2>Kursevaluering</h2>
 <p>Tast inn kurskode, og skriv en kort kommentar om hva du syns om kurset. </p>
 <form action="http://stud-vevtjener/kursevaluering.asp" method="post">
  <input type="hidden" name="tiladr" value="jarle.havik@hisf.no">
  <table>
   <tr><th colspan="2" align="left">Tilbakemeldingsskjema</th></tr>
   <tr><td colspan="2">Kurskode + studium + semester</td>
    <td><select name="Kurs">
         <option>BD640
         <option>DA620 - 1IBH
         <option>DA620 - 2IBH 2.semester
         <option>DA620 - 2IBH 4.semester
     </select>
    </td>
   </tr>       </table>
  <table>
   <tr><th colspan="5" align="left">Jeg synes kurset var:</th></tr>
   <tr>  <td>Lærerrikt :</td>
         <td colspan="4"> <input type="radio"name="larerikt" value="0">Helt uenig
         <input type="radio"name="larerikt" value="1">Uenig
         <input type="radio"name="larerikt" value="2">Delvis enig
         <input type="radio"name="larerikt" value="3">Enig
         <input type="radio"name="larerikt" value="4">Helt enig
    </td>
   </tr>
   :
   : Du kan fylle på med flere spørsmål - bruk fantasien. Kopier bare stammen over
   :
  </table>
  <p>Har du noen forslag til endringer?</p>
  <p><textarea name="kommentar" cols="60" rows=10>Skriv her...</textarea></p>
  <p><input type="submit" value="Send tilbakemelding">
     <input type="reset" value="Tøm skjema">
  </p>
 </form>
</body>
```

skript0808.html: Eksempel påtilbakemeldingsskjema med flerlinjet tekstfel.

Skriv inn koden og arkiver denne som skript0808.html.

Legg merke til at strukturen for å lage flere "avtrykkings-spørsmål" er relativt enkelt ved at man raskt kan kopiere koden for den første radio-knappen.



Figur 8.10. Eksempel på flerlinjet tekstfelt.

Prøv dette ut i praksis og legg merke til hvilke muligheter du har i tekst-området for å endre teksten.

Bruk også scriptet "kursevaluering.asp" som angitt i skjemaets action="" og legg merke til hvordan utskriften blir.

8.7.7 Mer om skjemaer

De elementene vi har omtalt så langt er de du vil få mest bruk for i arbeidet med skjemaer. HTML 4.0 har en del andre elementer som jeg har valgt å ikke ta med her, ettersom jeg mener de ikke er like relevante, samt at de lett gjør ting mer komplisert for folk som ikke har brukt skjemaer mye fra før av. Det er også verd å merke seg at nettleser-støtten for en del ting i HTML 4.0 er dårlig, hvilket medfører at enkelte skjema-elementer ikke er støttet i de nettleserne folk faktisk bruker.

Dersom du ønsker å vite mer om skjemaer, anbefaler vi at du leser gjennom HTML 4.0-spesifikasjonens kapittel 17, "Forms". Der vil du finne all spesifikasjon for HTML-delen av skjemaene. For server-delen vil det være nødvendig å lære seg et programmerings-språk, gjeme PHP eller ASP. Til dette finnes det flere relevante lærebøker tilgiengelig i bokhandelen med godt utvalg av data-bøker.

Oppgaver til kapittelene:

Oppgaver til kapittel 1 og 2.

1. Gjør kort rede for grunnideen bak WWW ?
2. Hva menes med strukturen i et dokument?
3. Nevn noen fordeler og ulemper ved bruk av WYSINWOG-programvare. Gjør også en kort drøfting av fordeler og ulemper ved bruk av tetkst-editorer.
4. Hvorfor bør man sjekke web-sider med flere nettlesere?
5. Hva er forskjellen mellom et *blokknivåelement* og et *tekstnivåelement*?
6. Skriv ned start- og slutt taggene, samt hvorvidt elementet er på blokk- eller teksnivå, for følgende elementer:
    a. Avsnitt
    b. Overskrift
    c. Lenke
    d. Adresse
    e. Mykt linjeskift
    f. Skillelinje
7. Hva bruker man en validator til?

Praktisk oppgave

Lag en web-side der du forteller om deg selv. Tenk på at det skal ha interesse for andre. Prøv også å ta i bruk de elementene du har lært til nå. Når du er ferdig kan du lagre den i mappen P:\htdocs\oppgaver, som oppgave01.html. Sjekk nå om den er tilgjengelig på netttet.

Oppgaver til kapittel 3.

1. Hvorfor er det viktig å bruke `alt = "tekst"` til bilder?
2. Nevn noen elementer vi kan benytte oss av for å manipulere tekst?
3. Hvorfor bør du bruke fysisk og logisk tekstmerking riktig?
4. Hva er en *entitet* i HTML sammenheng, og når brukes slike?
5. Gjør kort rede for de forskjellige listetypene som finnes.
6. Sett opp en oversikt over start- og slutt-tagger for listene i forrige eksempel, samt eventuelle andre tagger du kan bruke i forbindelse med disse.
7. Hva er start- og slutt-taggene for et sitat?
8. Hvilke byggeklosser har tabeller?

Praktisk oppgave

Lag en ny web-side hvor du tar i bruk elementene du skal ha lært til nå i kapittel 1 – 3, og som omhandler et av temaene nedenfor:

I. En web-side som forteller litt om hvordan man søker etter informasjon på Internett og som inneholder nyttige lenker for dette.

II. En web-side der du redegjør for dine tanker omkring bruken av Internett i studiesammenheng.

III. En web-side som presenterer annet av interesse for deg.

Når du er ferdig arkiverer den i mappen P:\htdocs\oppgaver, som oppgave01.html samtidig som du sender en melding til medstudenter og faglærer som inneholder URL'en til disse sidene.

Oppgaver til kapittel 4:

1. Skriv ned start- og slutt- taggene, samt hvorvidt elementet er på blokk- eller tekstnivå, for følgende elementer:
    a. Subskript
    b. Superskript
    c. Preformattert tekst
2. Hva er det viktig å huske på når man bruker preformattert tekst?
3. Forklar kort hvordan du vil gå frem for å sentrere et bilde, og hvorfor man må gjøre det på denne måten.
4. Hvordan lenker man direkte inn i et dokument?
5. Forklar hvordan man får til å bruke et bilde som en lenke?
6. Forklar kort hva *meta*-taggen brukes til.

Praktisk oppgave:

Denne oppgaven skal løses i samarbeid med en medstudent, og du er selv ansvarlig for å finne samarbeidspartner. Sammen skal dere lage en ny web-side, hvor dere tar i bruk elementene dere har lært så langt. Dere står fritt til valg av tema for web-sidene.

Oppgaver til kapittel 5 og 6

1. Hvordan kan man gi en tabell en kort overskrift eller underskrift?
2. Hvilke attributter lar oss koble sammen tabell-celler?
3. Med hvilke attributter kan vi påvirke en tabells utseende, og hva gjør de enkelte attributtene?
4. Kan bredden på en tabell eller en tabell-celle bestemmes, og i så fall hvordan?
5. Hvordan fungerer rammer (frames)?
6. Hvordan kan vi gjøre informasjonen i et rammesett tilgjengelig for de som ikke har en browser som støtter rammer?
7. Hvilke attributter trengs for at en lenke skal kunne åpnes i en spesiell ramme?

8. Kan vi på noen måte spesifisere språket for et dokument? Hva med innholdet av et enkelt element?

9. Hvilke fordeler og ulemper har stilsett?

Praktisk oppgave

Ta for deg web-sidene du lagde til kapittel 3, og ta i bruk de nye mulighetene i HTML 4.0 der det er mulig. Bruk gjerne rammer for å gi brukeren navigasjonsmuligheter.

Oppgaver til kapittel 8

1. Gjør kort rede for felttypene vi kan benytte for inntasting i skjemaer?

2. Forklar forskjellen mellom <input type=”checkbox”> og <input type=”radio”>?

3. Hvilke bruksområder har vi for skjulte felt?

4. Hvordan fungerer menyer, og hvilke muligheter gir disse?