



TOM HEINE NÄTT
EVA M. HORNNES
JOSTEIN NORDENGEN

Webutvikling

DATASERVICE
.NET

GYLDENDAL
UNDERVISNING

Dette heftet inneholder de 13 første kapitlene av boka «Webutvikling». Heftet tillates kopiert til elever og studenter for bruk i skoleåret 2016/2017.

Boka er under utvikling, og en mer omfattende versjon av boka vil bli lagt ut på våre nettsider utover høsten. Ferdig bok er ventet våren 2017.

Halden 2. november 2016

Dataservice as/Gyldendal Undervisning



www.it-1.no

Innholdsfortegnelse

{01} Introduksjon til webutvikling s.10

- Utviklingsmiljø **11**
- Ditt første prosjekt **11**
- Lage nettsiden med HTML **12**
- Knytte til et stilark **12**
- Lage side nummer to **14**
- Koble hovedside og side nummer to **15**
- Lagring av nettsider **16**
- Skrive god kode **19**
 - EKSEMPEL - Lage en presentasjon av deg selv **20**

{02} HTML og CSS s.23

- HTML **23**
 - Strukturen **24**
 - Malen **24**
 - Lagre malen **25**
- CSS **26**
 - Strukturen **26**
 - Forholdet mellom CSS og HTML **27**
 - Forsøk selv **27**
 - Attributtene id og class **28**
 - Forsøk selv **29**
 - Navngiving av klasser og ID-er **29**
 - Multiple class **30**
 - Forsøk selv: **32**
 - Taggene og <div> **32**
 - Forsøk selv: **33**
 - CSS Box model **34**
 - Forsøk selv **35**
 - Ulike nivåer av CSS **36**
 - Cascading **37**
 - Forsøk selv: **38**
 - De to ekstra CSS-nivåene **39**
 - Validering av nettsider **40**
 - Kompatibilitet **41**
 - Når skal man benytte hvilken metode? **44**
 - Kommentarer **44**
 - Mer informasjon om HTML og CSS **46**
 - W3C og WHATWG **46**
 - W3Schools **47**
 - Google **47**
 - StackOverflow **47**
 - Andre nettsiders kode **47**

{03} Strukturere websider s.48

- Formatering av tekst **48**
 - Paragrafer **49**
 - Forsøk selv: **50**
 - Overskrifter **50**
 - Utheving og redigering **51**
 - Forhåndsformatert tekst **52**
 - Forsøk selv: **52**
- Tegnsett **53**
- Bilder **54**
 - Bildeformater **55**
 - JPG (JPEG) **55**
 - GIF **55**
 - PNG **55**
 - Størrelser **56**
 - Figurer **57**
 - Forsøk selv: **58**
- Linker **59**
 - Forsøk selv: **60**
 - Internlinker **60**
- Tabeller **61**
 - Forsøk selv: **62**
 - Tabellceller som dekker flere rader/kolonner **63**
 - Forsøk selv: **64**
 - Lister **64**
 - Forsøk selv: **65**
 - Nestede lister **65**
 - Forsøk selv: **66**
 - Definisjonslister **66**
 - EKSEMPEL - Lage en timeplan **67**
- Medieinnhold **70**
 - Lyd **70**
 - Video **72**
 - Forsøk selv: **73**
 - Video og CSS **74**
- Semantiske tagger **75**
 - <main> **75**
 - <section> og <article> **75**
 - <header> og <footer> **76**
 - <nav> **76**
 - <aside> **76**
 - Hvorfor benytte tagger som ikke har noen visuell effekt? **76**

{04} Formgi nettsider s.78

Selectorer **79**
Sammensatte selectorer **79**
Relasjoner og selectorer **80**
Forsøk selv: **81**
Selectorer på attributter **82**
Forsøk selv: **83**
Pseudoelementer **83**
Pseudoklasser **85**
Forsøk selv: **87**
EKSEMPEL - interaktivitet **87**
Måleenheter **90**
Farger **91**
Forsøk selv: **92**
Tekst **93**
Skrifttyper **95**
Forsøk selv: **98**
CSS Bilder **98**
EKSEMPEL - Fornøyelsespark **100**
Mer om CSS-bokser **102**
Kantlinjer **102**
Forsøk selv: **103**
Marger **104**
Skygge **105**
Forsøk selv: **106**
Størrelse og overflow **106**
Forsøk selv: **108**
Visningsmetoder **108**
Nettleserprefiks **109**

{05} Mer om layout og design s.110

Posisjonering **110**
Relativ posisjonering **111**
Forsøk selv: **112**
Låst posisjonering **113**
Absolutt posisjonering **114**
Forsøk selv: **115**
Lagjustering **116**
Utfordringer med posisjonering **116**
Floating **117**
Flexbox **119**
Forsøk selv: **122**
Flexbox og layout i flere dimensjoner. **124**
Noen vanlige layout-oppsett **126**
Forsøk selv: **129**
Menyer **130**
Forsøk selv: **133**
Medietyper **134**
Medietyper i samme stilark **135**
Spesielle CSS funksjoner for utskrift **135**

Noen gode råd for å lage utskriftsvennlige
design **137**
Media queries **137**
Responsive web design **139**
Skalere bilder **141**
EKSEMPEL - Responsive design **142**

{06} Eksempel - Festival s.146

EKSEMPEL - Festival **146**
Lage menyen **148**
Lage programside **157**

{07} Publisering s.163

Søkemotoroptimalisering **164**
Nettsiders struktur **164**
Tilleggsinformasjon **164**
robots.txt **164**
XML sitemap **165**
Analyse **166**
Google Webmaster Tools **166**
Google Analytics **168**
Sunn skepsis **170**
Microdata **170**
Rich snippets **174**
Verktøy for Microdata **175**
Kobling mot sosiale medier **178**
Optimalisere for deling **178**
Facebook **178**
Twitter **179**
Google+ **180**
Pinterest **180**
Inkludere data fra sosiale medier **180**
Facebook **180**
Twitter **181**
Tilgjengelighet **182**
Syntetisk tale **183**
Annet **184**
RSS **184**
Nettsideicon **186**

{08} Hurtigreferanse HTML/CSS s.187

HTML **188**
Grunnstruktur **188**
Formatering av tekst **188**
Escapetegn **188**
Bilder **188**
Linker **189**
Internlinker **189**
Medieinnhold **191**
Lyd **191**
Attributter **192**
Video **192**
Attributter **192**
Semantiske tagger **193**
Nettsideicon **193**
CSS **193**
Kobling mellom HTML og CSS **193**
Grunnstruktur **194**
Box model **194**
Kantlinjer **194**
Selectorer **195**
Attributtverdier **196**
Pseudoelementer **196**
Pseudoklasser **196**
Måleenheter **197**
Absolutte **197**
Relative **197**
Fargeangivelser **198**
Justere tekst **198**
Fonter **198**
Webfonter **199**
CSS-bilder **199**
Tabeller **199**
Posisjonering **200**
Relativ **200**
Fixed **200**
Absolute **200**
Lagjustering **200**
Floating **200**
Flexbox **201**
Medietyper **202**
Media queries **202**
Skalere bilder med CSS og media queries **203**

{09} Introduksjon til dynamiske nettsider s.204

Statiske og dynamiske nettsider **204**
WampServer **206**
Installasjon av WampServer **207**
Starte WampServer **210**
Feil under oppstart av WampServer **212**
Avslutte IIS **213**
Avslutte andre MySQL-instanser **214**
Andre applikasjoner som opptar port 80 **214**
Teste WampServer **216**
MAMP **216**

{10} Din første dynamiske nettside s.218

Opprette den dynamiske nettsiden **218**
Ting å huske på når du lager dynamiske nettsider **220**
Mer om visning av dato og tid **220**
Lære mer om PHP-koder **221**

{11} Arbeide med MySQL s.222

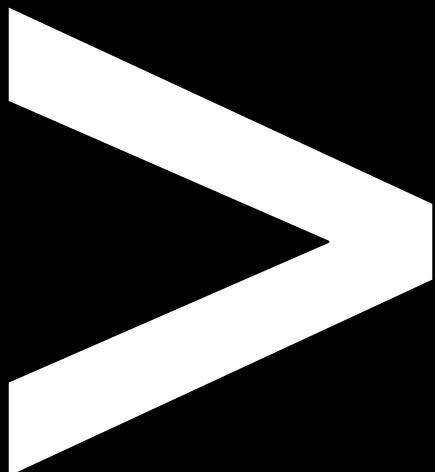
Brukergrensesnitt til MySQL **222**
Opprette tilkobling til MySQL i Workbench **224**
Opprette tilkobling til MySQL i MAMP **225**
Koble til MySQL fra Workbench **226**
Opprette et nytt prosjekt **226**
Opprette databasetabeller **228**
Legge inn data i databasetabeller **230**
Flere databasetabeller **231**
Slette rader **234**
Opprette brukere **234**
Legge til brukere og rettigheter **235**
phpMyAdmin **237**
Datatyper i MySQL **238**

{12} Koble dynamiske nettsider og MySQL s.239

Koble til en database 240
Hente ut en tabelloversikt av dataene 241

{13} Hente ut data s.243

Koble flere tabeller 243
Presentere data ved hjelp av regioner 245
Tilpassede datapresentasjoner 247
Søk i data 248
Mer om skjemaer 250
Tekstbokser 250
Tekstområder 253
Nedtrekkslister 253
Knapper 254



1 Introduksjon til webutvikling

I dette kapitlet vil du lære

- om hva HTML og CSS er
- å lage din første nettside
- om lagring av nettsider
- å skrive god kode

Du har kanskje alt hørt om begrepene HTML og CSS (også kalt stilark). Dette er de to teknikkene/språkene som benyttes for å lage en nettsider. Det er ikke helt riktig å kalle dette for programmeringsspråk, slik mange gjør, for det er nettsider og ikke programmer vi skal lage foreløpig. HTML er en forkortelse for *Hyper Text Markup Language*, mens CSS er en forkortelse for *Cascading Style Sheet*. Rent formelt er HTML et *markeringsspråk* (markupspråk) og CSS et *deklarativt språk*, men du trenger ikke bekymre deg for dette nå.



Vi kommer tilbake til en mer grundig forklaring på hva HTML og CSS er i neste kapittel, men foreløpig kan vi si så mye som at HTML er et språk for å markere hva slags betydning og struktur innhold skal ha. CSS derimot er et språk for å definere hvordan innhold skal presenteres, eller med andre ord det vi ofte omtaler som design.

Vi skal starte det første kapittelet i denne boka med et «kom i gang»-eksempel og litt praktisk informasjon om det å lage nettsider.

Utviklingsmiljø

En av fordelene med HTML og CSS er at de ikke er knyttet opp mot noe bestemt utviklingsmiljø eller spesiell editor der vi skriver kodene. Så lenge du har en teksteditor som lagrer *ren tekst*, kan den benyttes til å lage nettsider med. Dette betyr f.eks. at Windows sin Notepad/Notisblokk kan benyttes.

Vi anbefaler imidlertid å arbeide med en noe mer avansert editor for å forenkle arbeidet med å lage nettsider. Fargelegging av kode (*såkalt syntax highlighting*) og muligheten til å arbeide med flere filer samtidig (faner) er en stor fordel. I tillegg kan det være greit at editoren viser linjenummerering.

Enkelte foretrekker også verktøy som gir forslag på kode (*såkalt code completion*) og som hjelper til med skrivingen, men vår anbefaling er å vente litt til med slike funksjoner. Vi anbefaler også å vente med verktøy der du kan designe nettsidene grafisk inntil du har lært deg grunnleggende koding.

I denne boka kommer vi til å benytte verktøyet *Notepad++* som kan lastes ned fra <http://notepad-plus-plus.org>. Du kan imidlertid fint benytte din favoritteditor i stedet. Andre muligheter er f. eks. *Brackets*, *WebStorm* og *Sublime Text*. For Mac OS vil også verktøy slik som *TextWrangler* og *Edit* kunne benyttes. Det er også mulig å benytte tyngre verktøy, slik som Dreamweaver, ettersom de fleste slike har en kodemodus.

For å se på resultatet trenger du kun en nettleser, slik som man forbinder med nettsider ellers. Du trenger ikke bekymre deg over at du må publisere nettsidene til en webserver ettersom en nettleser også fint kan vise lokale filer.

I denne boka kommer vi til å benytte *Google Chrome* som nettleser, men du står fritt til å benytte en annen. I teorien skal alle nettlesere vise en nettside likt, men dette er dessverre ikke alltid praksis. I tillegg kommer skermopløsning inn som en faktor. Følger du eksemplene i boka kan dine nettsider av disse grunner ha noen minimale forskjeller i utseende i forhold til skjermbilder i denne boka.

Ditt første prosjekt

La oss nå lage vår første svært enkle prosjekt for å komme i gang. Vi ønsker her å vise deg litt av arbeidsflyten og hvordan du kommer i gang. Ikke bekymre deg så mye over selve kodene, for de kommer vi tilbake til med mer grundige forklaringer i neste kapittel.

Lage nettsiden med HTML

- Opprett en mappe som du gir navnet webutvikling. I denne mappa lager du en mappe kalt kapittel1. Lagre alle filer til dette prosjektet i denne mappa.

- Start ++ (eller tilsvarende kodeeditor) og skriv inn følgende tekst:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Vår første nettside</title>
        <meta charset="utf-8" />
    </head>

    <body>
        <h1>Velkommen til vår første nettside</h1>
        <p>Mer innhold vil komme etterhvert</p>
    </body>
</html>
```

- Lagre dokumentet i mappen *kapittel1* med filnavnet *index.html*
- Dobbeltklikk på fila i filutforskeren for å se på resultatet i nettleseren din
Du kan alternativt velge **File - Open** i nettleseren.



Mens man utvikler nettsider er det vanlig å arbeide med de *lokalt*. Det vil si at vi ikke flytter de over på en webserver hver gang vi skal se hvordan de ser ut. Publisere nettsidene til en webserver gjør vi vanligvis først når vi er ferdige.

Når vi kun har nettsidene lokalt, vil URL-en starte med *file://* i stedet for *http://*.

Det er nok foreløpig lettere å åpne fila på den måten vi skisserte, enn å skrive seg frem til denne URL-en.

Knytte til et stilark

For å endre utseende til nettsiden skal vi knytte til et stilark (CSS). Også stilark kommer vi tilbake til i neste kapittel, så dette blir bare en liten introduksjon. Alt som går på innhold skal plasseres i en HTML-fil, mens alt som går på utseende og design skal gjøres via CSS.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Ettersom nettleseren åpner HTML-fila, må vi fortelle i denne fila hvilket stilark som vi ønsker å benytte for utseende.

1. Velg **Ny > Fil** i din kodeeditor og skriv inn følgende kode:

```
body {  
    background-color: lightgray;  
    font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;  
}  
  
h1 {  
    text-decoration: underline;  
}
```

2. Lagre fila med navnet *stil.css* i mappen kapittel1
3. Sørg for at fila *index.html* er åpen i editoren
4. Legg til følgende **<link>**-tagg i **<head>**-delen av HTML-fila:

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Vår første nettside</title>  
        <meta charset="utf-8" />  
        <link rel="stylesheet" type="text/css" href="stil.css" />  
    </head>  
  
    <body>  
        <h1>Velkommen til vår første nettside</h1>  
        <p>Mer innhold vil komme etter hvert.</p>  
    </body>  
</html>
```

5. Test nettsiden



Som du ser benytter nå nettsiden stilarkets regler for hvordan **<body>** og **<h1>**-taggene med innhold skal presenteres. Legg spesielt merke til at siden vår **<h1>**-tagg står inne i **<body>**-taggen i HTML-dokumentet, så vil også **<h1>**-taggen få egenskapene vi definerer for **<body>**.

Lage side nummer to

Vi skal nå utvide eksempelet med en ekstra nettside, som også inneholder et bilde.

1. Finn et bilde på disken din eller Internett av typen JPG og lagre dette i mappen *kapittel1* med filnavnet *test.jpg*. Du kan også finne et bilde blant ressursene til boka
2. Åpne et nytt dokument i Notepad++ (eller tilsvarende kodeeditor)
3. Skriv inn følgende kode

```
<!DOCTYPE html>
<html>
    <head>
        <title>Neste side</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="stil.css" />
    </head>

    <body>
        <h1>Dette er side nummer to</h1>
        
    </body>
</html>
```

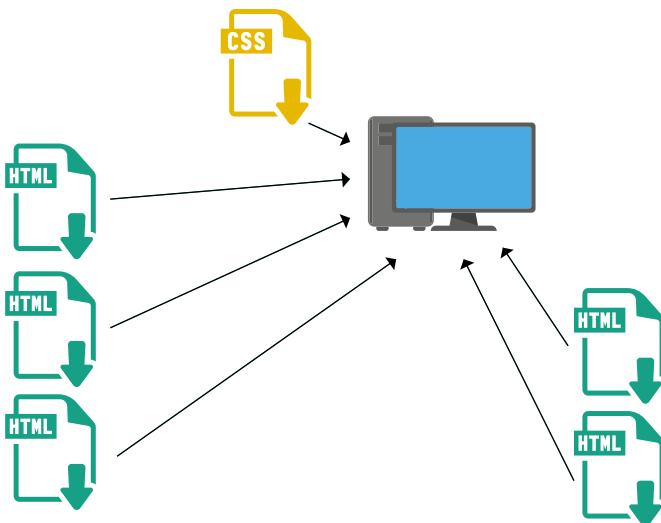
4. Lagre denne nettsiden med filnavnet *side2.html*, i samme mappe som *index.html* og *test.jpg*
5. Test fila *side2.html* i nettleseren din. Avhengig av størrelsen på bildet du satt inn, kan bildet bli mindre eller større enn det som vises hos oss.

Vi skal lære å sette størrelsen på bilder etterhvert



Som du ser er det meste av koden til *side2.html* svært likt koden til fila *index.html*. Legg spesielt merke til at vi også her kobler inn stilarket *stil.css*, og at også denne

nettsiden følger reglene angitt i denne. Dette er en av de store styrkene med stilark som egne filer. Vi kan i én enkelt CSS-fil fortelle hvordan utseende på nettsider skal være, og så koble dette mot et ubegrenset antall HTML-filer.



I tillegg introduserer vi taggen `` som er taggen vi benytter for å sette inn bilder i en nettside. I denne taggen må vi angi hvilken bildefil vi ønsker vise ved hjelp av attributtet/egenskapen `src` (forkortelse for `source`). I tillegg må vi alltid angi en *alternativ tekst* som skal vises hvis bildet ikke kunne lastes eller bildevisning ikke er tilgjengelig. Dette gjøres i attributtet `alt`.

Koble hovedside og side nummer to

Siste steg i dette eksempelet består i å lage en kobling (link) fra `index.html` til `side2.html`.

1. Åpne opp kildekoden til nettsiden `index.html`, og legg inn en link/kobling til `side2.html` der:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Vår første nettside</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="stil.css" />
    </head>

    <body>
        <h1>Velkommen til vår første nettside</h1>
        <p>Mer innhold vil komme etter hvert.</p>
        <p>Du kan også gå til <a href="side2.html">neste side</a>.</p>
    </body>
</html>
```

2. Test nå at du får opp en link på førstesiden som gjør at du kan navigere deg til den andre siden



Det er taggen `<a>` (forkortelse for *anchor/anker*) som oppretter det vi kjenner som linker. Denne taggen har et attributt kalt `href` (forkortelse for *hyper reference/hyperreferanse*) som forteller hvilken fil eller nettadresse det skal linkes til. Mellom `<a>` og `` setter vi det som skal være klikkbart, eller med andre ord *linkteksten*.

Som sagt kommer vi tilbake til mer utdypende forklaringer av HTML og CSS i neste kapittel. Foreløpig er det viktigste at du forstår hvor du plasserer kodene og hvordan du tester resultatet i en nettleser.

Klarer du å legge inn en tilsvarende link i fila side2.html slik at man kan hoppe tilbake til index.html?

Lagring av nettsider

Lagring av filer høres ganske så enkelt ut, men det er et par ekstra hensyn vi må og bør ta når vi arbeider med nettsider.

Det er bl.a. visse begrensninger du må passe på når du skal velge filnavn til et HTML- eller CSS-dokument. Du bør unngå bruk av norske spesialtegn (æ, ø og å). Unngå også å bruke mellomrom og spesialtegn som punktum, komma, semikolon og lignende.

Vi anbefaler også at du alltid kun bruker små bokstaver i filnavnet ettersom mange webservere er såkalt *case sensitive* (dvs. små og store bokstaver er ikke samme tegn). En vanlig feil er f.eks. å referere til fila *Stil.css*, mens fila egentlig heter *stil.css*:

```
<link rel="stylesheet" type="text/css" href="Stil.css" />
```

Når du tester lokalt på en Windows-maskin (som ikke skiller på dette) vil alt fungere, og i det du flytter nettsiden over til en webserver (som ofte kjører Linux) vil ikke stilarket bli funnet.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Hvor mange tegn du kan ha i filnavnet, er avhengig av hvilken type webserver du skal publisere nettsidene til. Det vil derfor være lurt å holde seg til maksimalt 255 tegn.

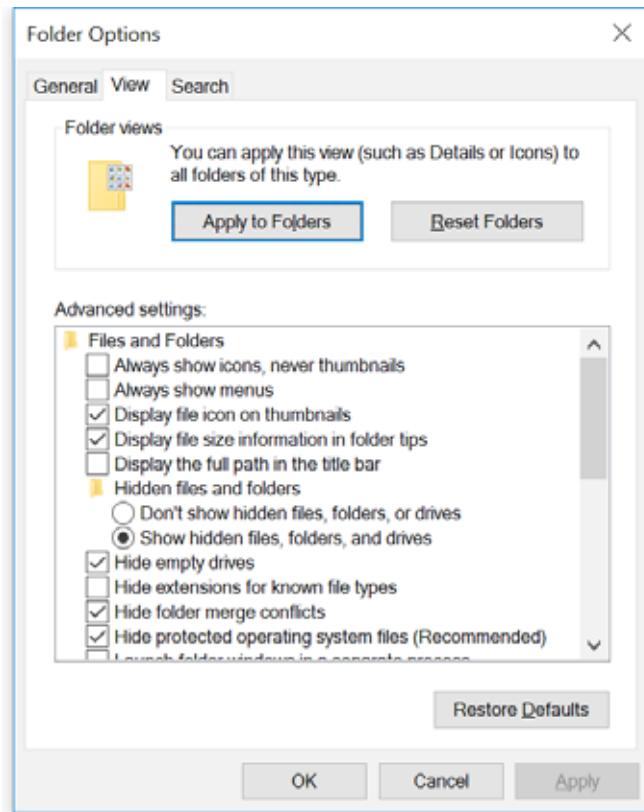
Alle nettsider må ha filendelsen *html* (med unntak av dynamiske nettsider som benytter *asp*, *php*, *jsp* og lignende). Dersom du ikke bruker denne filendelsen, vil det ofte ikke være mulig å vise nettsiden i en nettleser.

Endelsen må være *html* og *css*. Enkelte editorer legger til f.eks. *.txt* etter filnavnet, slik at det blir *index.html.txt* eller *stil.css.txt* om du ikke velger *Alle formater* eller *HTML/CSS* under lagring.



Når du arbeider med utvikling av nettsider kan det være lurt å få operativsystemet til å vise filendeler i filutforskeren. Standard innstilling i Windows er at filendelsene ikke vises. For å unngå unødvendige problem ved at du refererer til feil filnavn, er det viktig å skru på visning av filetternavn.

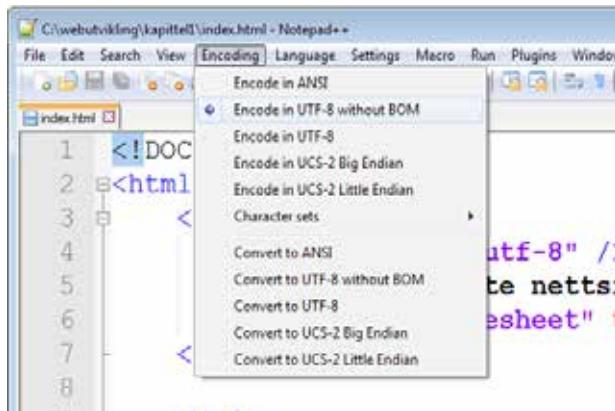
Åpne Windows Utforsker og velg **Fil - Endre mappe- og søkealternativer (File - Change folder and search options)** videre velger du fanen **vis / view** og fjerner markeringen for punktet **Skjul filetternavn for kjente filtyper (Hide extensions for known file types)**



TIPS

Enkelte eldre nettsider og webservere opererer fortsatt med filendelsen *htm* (altså uten den siste *l*-en) fra tiden da Windows kun taklet filetternavn på tre tegn. Dette er imidlertid frarådet å benytte nå.

Det er også viktig at du lagrer fila med tegnsettet UTF-8 eller *Unicode* valgt. Noen editorer har valg for dette i lagre-dialogen, mens andre har det som menyvalg i selve applikasjonen. De fleste nyere editorer og operativsystemer har imidlertid UTF-8 satt som standard. I Notepad++ finner du valget i menyen *Encoding*.



Skal du lage en startside, hovedside eller førsteside i et webområde, bør den alltid ha filnavnet *index.html*.

Årsaken til at dette må gjøres slik, er at du skal slippe å skrive hele nettadressen til førstesiden i nettleseren. Skal du for eksempel gå til Gyldendals hjemmeside, behøver du ikke å skrive <http://www.gyldental.no/index.html>. Det er nok å skrive <http://www.gyldental.no>. Nettsiden *index.html* vises automatisk når denne adressen skrives. Dette styres av webserveren som webområdet er lagret på.

TIPS

Enkelte webservere benytter også *default.html* i stedet for eller i tillegg til *index.html* som forventet filnavn på hovedsiden.

MERK

Det kan være lurt å unngå spesielle systemfoldere slik som *skrivebord* og *mine dokumenter* når du skal lagre filer for webprosjekter.

Skrive god kode

HTML og CSS er svært lite nøye på hvordan koden er formatert og strukturert.

Strengt tatt kan du skrive all koden på en linje, slik som

```
<!DOCTYPE html><html><head><title>Vår første nettside</title><meta charset="utf-8" /></head><body><h1>Velkommen til vår første nettside</h1><p>Mer innhold vil komme etterhvert</p></body></html>
```

eller

```
body{background-color:lightgray;font-family:'Arial, 'Helvetica Neue', Helvetica, sans-serif;}{text-decoration:underline;}
```

Det er derimot veldig viktig for oss som utviklere av nettsider å skrive kode som er «pent» strukturert. Du kan jo se forskjellen på lesbarheten i de kodebitene vi nettopp presenterte, og koden til eksemplet vi gjennomgikk tidligere.

Den gylnde regelen for HTML er at hver gang vi starter en tag, rykker vi inn all påfølgende kode ett hakk. Dette kalles å *indentere* koden. Du velger selv om du vil benytte tabulator eller 3-4 mellomrom. Når vi skriver slutt-taggen stopper vi innrykket igjen. Dette kan gjelde på mange nivåer, da vi kan starte tagger som er såkalt *nestede* (en tagg inne i en annen tagg).

Koden over vil se slik ut om vi formaterer den riktig.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vår første nettside</title>
    <meta charset="utf-8" />
  </head>

  <body>
    <h1>Velkommen til vår første nettside</h1>
    <p>Mer innhold vil komme etterhvert</p>
  </body>
</html>
```

Tilsvarende har vi regler vi bør forholde oss til når vi skriver CSS kode. Her er det vanlig å rykke alle egenskaper inn ett nivå.

```
body {
  background-color: lightgray;
  font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;
}

h1 {
  text-decoration: underline;
}
```

Det kan virke svært pirkete å være så nøye med dette når nettleseren ser totalt bort i fra det, og så lenge prosjektene er enkle og oversiktlige. Det er imidlertid nå som du er i startfasen det er lett å få en god vane. Gjør du det ikke riktig nå er det svært vanskelig å få denne gode vanen når du kommer over i større prosjekt som trenger en god formatering for å ha oversikt.

EKSEMPEL - Lage en presentasjon av deg selv

I dette eksemplet skal du lage en nettside med en presentasjon av deg selv. I tillegg skal du lage en nettside som gir en presentasjon av din største fritidsinteresse. Du skal også lage en link mellom de to sidene.

1. Opprett en mappe i mappen *kapittel1* som du gir navnet presentasjon. Alle filene du lager i dette eksemplet skal lagres i denne mappa
2. Finn et bilde av deg selv og lagre det i mappa som meg.jpg
3. Start Notepad++ (eller tilsvarende kodeeditor) og skriv inn følgende tekst:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Presentasjon av **NN**</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="stil.css" />
    </head>
    <body>
        <h1>Presentasjon av **NN**</h1>
        
        <h2>Personlig informasjon</h2>
        <p>**personlig info**</p>

        <h2>Hjemstedet mitt</h2>
        <p>**info om hjemsted**</p>

        <h2>Fritidsinteresser</h2>
        <p>**fritidsintresser**</p>
        <a href="fritid.html">Les mer om **interesse**</a>

        <hr>
        <p>&copy; **NN**</p>
    </body>
</html>
```

4. Lagre dokumentet i mappa med filnavnet *index.html*
5. Dobbeltklikk på fila i filutforskeren for å se på resultatet i nettleseren din. Du kan alternativt velge **Fil > Åpne** i nettleseren
6. Legg merke til at vi refererer til en CSS-fil i koden, men vi har ennå ikke lagd den. Velg Fil > Ny i din kodeeditor og skriv inn følgende (Vi skal se nærmere på de ulike CSS-kodene senere i boka):

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
body {  
    background-color: lightgray;  
    font-family: sans-serif;  
    margin-left: 20px;  
}  
  
img {  
    width: 200px;  
    border: solid 1px white;  
    box-shadow: 5px 5px 20px gray;  
}  
  
h1,h2 {  
    color: white;  
    text-shadow: 0px 0px 8px black  
}  
  
h1 {  
    font-size: 30pt;  
}
```

7. Lagre fila med navnet *stil.css* i mappa
8. Last inn siden (*index.html*) på nytt i nettleseren og sjekk ut hvordan siden er blitt endret etter at vi lagde CSS-fila:



Personlig informasjon

personlig info

Hjemstedet mitt

info om hjemsted

Fritidsinteresser

fritidsinteresser

[Les mer om **interesse**](#)

© **NN**



The screenshot shows a web page with the title "Presentasjon av **NN**". It features a profile picture of a young woman with red hair. Below the picture, there are three main sections: "Personlig informasjon" (Personal information) containing the placeholder text "**personlig info**"; "Hjemstedet mitt" (Homeplace) containing the placeholder text "**info om hjemsted**"; and "Fritidsinteresser" (Hobbies) containing the placeholder text "**fritidsinteresser**" and a link "[Les mer om **interesse**](#)". At the bottom right, it says "© **NN**".

9. Gå i koden og endre **NN** til ditt eget navn. Endre **personlig info** slik at det står fullt navn, hvor gammel du er, hvilken skole du går på osv. Endre **info om hjemsted** slik at det står en kort beskrivelse av stedet der du bor. Endre **fritidsinteresser* slik at det står noe om de ulike fritidsinteressene du har. Endre **interesse** til navnet på din favoritt fritidsinteresse, f.eks. håndball, Game of Thrones eller dataspill. Sjekk ut resultatet i nettleseren
10. Lag en egen nettside (HTML-fil) som forteller litt om din favoritt fritidsinteresse. Ta gjerne en kopi av *index.html* og endre på denne. Pass på at du lagrer fila som *fritid.html* slik at linken (<a>-taggen) i *index.html* virker. Hvis du ønsker et helt annet utseende på denne sida kan du lage en egen CSS-fil. Hvis du gjør dette må du passe på å forandre filnavnet i link taggen i HTML-fila slik at den viser til den nye CSS-fila
11. Hvis du ønsker kan du lage flere sider med informasjon, f.eks. en side som forteller litt om din favoritt film, din favoritt artist osv. Pass på at du linker til sidene i *index.html* ved å lage flere <a>-tagger



2 HTML og CSS

I dette kapitlet vil du lære

- om den grunnleggende HTML-strukturen
- om den grunnleggende CSS-strukturen
- om ulike koblinger mellom HTML og CSS
- om CSS sin box model
- om ulike nivåer av CSS
- om validering av HTML og CSS
- om kompatibilitet
- om relative og absolutte referanser
- om kommentering
- hvor du finner mer informasjon

I forrige kapittel fikk du en rask start på det å lage nettsider med HTML og CSS. I dette kapittelet vil vi ta for oss en litt grundigere forklaring av hvordan HTML og CSS er bygget opp. Vi vil også se på hvordan HTML og CSS knyttes sammen.

HTML

HTML er et såkalt *markeringsspråk* (eng: *markup language*).

Ideen er at man har et forholdsvis ordinært tekstdokument hvor tekst markeres med såkalte tagger (markører) for å gi denne teksten spesiell betydning eller egenskaper.

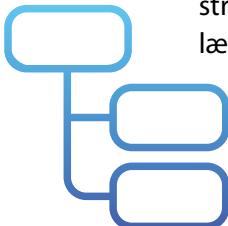
Selv navnet på språket, HTML, står for *HyperText Markup Language*. Begrepet *HyperText* refererer blant annet til det vi kjenner fra Internett som klikkbare lenker som er idéen om at en del av den ordinære teksten kan bringe oss videre til andre nettsider. *Markup Language* refererer til markeringen av tekst med tagger for å gi den en spesiell betydning.



Strukturen

Det er i all hovedsak to ting vi trenger vite om HTML-koding. Vi må vite hvordan strukturen i kodespråket er bygget opp, og hvilke kodeord/tagger som er tilgjengelig. Dette minner litt om et normalt språk der vi har grammatikk og et vokabular.

Ekspelet vi så på i forrige kapittel inneholder alt du trenger å vite om selve strukturen. En oversikt over de viktigste kodeordene er imidlertid mer omfattende å lære, men vi skal i løpet av de neste kapitlene ta deg gjennom en stor del av disse.



Det finnes i praksis to ulike typer tagger i HTML. Den ene typen tagger er de som markerer et område med tekst og gir denne teksten en spesiell betydning. Disse taggene starter og avslutter henholdsvis før og etter teksten. Den avsluttende taggen inneholder da en / for å markere avslutningen.

Et eksempel på dette er f.eks. ``-taggen (for viktig tekst):

Dette er et `viktig ord` i teksten.

Den andre formen for tagger er de som ikke markerer et område, men bare et punkt i teksten. Disse både starter og avslutter samtidig. Dette markeres ofte ved at taggen ender med en /, men dette er ikke noe absolutt krav i standard HTML5.

Typisk eksempel på dette er linjebrudd-markeringen:

Her er første linje.`
`Her er andre linje.

Taggene kan også ha attributter (eller egenskaper om du vil) som forteller noe om hvordan taggen skal oppføre seg. Et eksempel er ``-taggen som setter inn et bilde, og som minimum har de to attributtene src (hvor bildefilen befinner seg) og alt (hva er den alternative tekstrepresentasjonen for bildet):

Her er et bilde: ``

Dette er alt du trenger å kunne om struktur av HTML-dokumenter. Med andre ord er HTML et svært enkelt kodespråk. Å få oversikt over taggene og attributtene tar imidlertid litt lengre tid.

Malen

Det minste HTML-dokumentet, og «malen» om du vil, ser slik ut:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stilarkfil.css" />
  </head>

  <body>
    <p>Innhold</p>
  </body>
</html>
```

Her har vi også tatt med en kobling til et stilark, selv om det ikke er påkrevd.

Som du ser starter alltid et HTML-dokument med `<!DOCTYPE html>`. Dette er egentlig ikke en del av selve HTML-koden, men en instruksjon til nettleseren som skal vise nettsiden om at formatet nettsiden er skrevet i er HTML.

Deretter er alle HTML-dokumenter bygget opp av seksjoner som har en start-tagg og en slutt-tagg. En av disse seksjonene er hele nettsiden som er markert med `<html>` og `</html>`. Denne seksjonen er så delt inn i to underseksjoner, der én er `<head></head>` og én er `<body></body>`.

I seksjonen `<head>` plasserer vi all informasjon om nettsiden. Dette er såkalt metadata, og vises som regel ikke direkte til sluttbrukeren. Unntaket er tittelen på nettsiden som markeres med `<title>` og `</title>`. Denne tittelen vises bl.a. i tittelfeltet på nettleservinduet eller som tittel på faner. Den benyttes også som forslag til tekst på bokmerker, og som en del av informasjonen som vises i søkemotorenes (f.eks. Google) resultatlister.

Seksjonen inneholder også en tagg som forteller hvilket tegnsett som nettsiden er laget med. Her er det i dag mest vanlig å benytte `utf-8` som tegnsett (attributtet `charset`).

Skal vi benytte stilark, er det også aktuelt å fortelle nettleseren at den skal linke inn en ekstern ressurs gjennom `<link>`-taggen. Her forteller vi ved attributtet `rel` at relasjonen er av typen `stylesheet`, og i attributtet `type` at det er `text/css` som er den eksakte typen. Vi angir også en referanse til selve fila ved hjelp av attributtet `href`.

Den andre seksjonen en nettside alltid har er `<body>`. Her plasserer vi alt som skal bli en synlig del av nettsiden, slik som tekst og bilde.

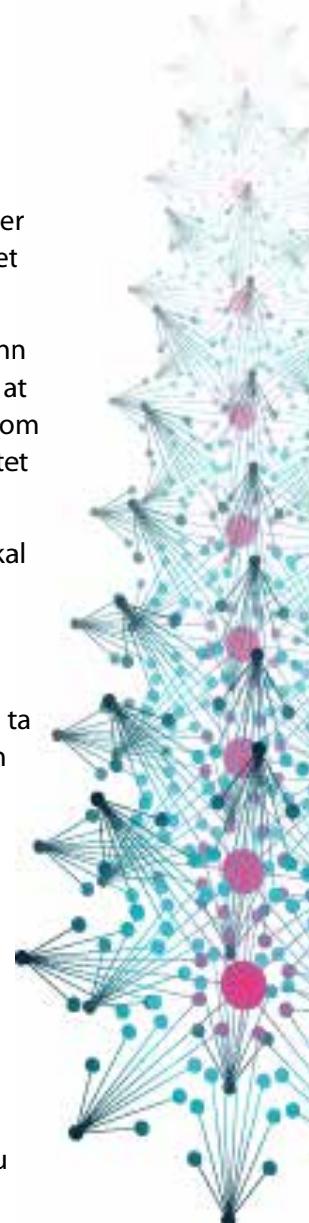
Lagre malen

Det kan være lurt å lagre malen for HTML-dokumenter i en egen fil, slik at du kan ta utgangspunkt i denne når du skal lage nettsider videre. Da slipper du å skrive inn grunnstrukturen hver eneste gang.

1. Opprett en ny fil i Notepad++ (eller valgte kodeeditor)
2. Skriv inn HTML-koden som vist i forrige seksjon
3. Lagre fila som `mal.html` i mappa `webutvikling` som du opprettet i forrige kapittel

For å benytte deg av malen gjør du følgende:

1. Åpne `mal.html` i Notepad++ (eller valgte kodeeditor).
2. Velg **Fil >Lagre som** og plasser kopien av fila med valgfritt filnavn der du ønsker å ha den
3. Begynn å legge til innhold og kode i malen på vanlig måte



CSS

CSS er laget for å fortelle hvordan noe skal se ut og definerer med andre ord et design. Selve forkortelsen CSS står for *Cascading Style Sheets*, og vi kommer tilbake til hva dette betyr i praksis senere i kapittelet. På norsk omtales ofte CSS som *stilark*.

Strukturen

CSS har på samme måte som HTML en slags grammatikk og et vokabular. Her er grammatikken enda enklere. I sin grunnform består CSS kun av en rekke *regler* for å beskrive utseendet. Hver regel er bygget opp av en *selector* som forteller hvilke elementer man ønsker å endre. Inne i regelblokken angir man et eller flere par med *egenskap* og *verdi*. Egenskapen skiller fra verdi med et kolon, og egenskap/verdi-par skiller med semikolon.

```
selector {  
    egenskap: verdi;  
    egenskap: verdi;  
    egenskap: verdi;  
}
```

Under ser du noen eksempler på praktisk bruk av selectorer og par med egenskap og tilhørende verdi.

Sett skrifttype (font) til **Arial**, samt fontstørrelse til **large** for alle paragrafer:

```
p {  
    font-family: Arial;  
    font-size: large;  
}
```

Angi at det skal være en 20 piksler bred marg rundt alle bilder:

```
img {  
    margin: 20px;  
}
```

All tekst i hele dokumentet skal vises i kun store bokstaver:

```
body {  
    text-transform: uppercase;  
}
```

Som du ser, sammenfaller det vi angir som selector med de ulike taggene i HTML. Etter hvert skal vi se at det også finnes andre varianter av selectorer, men foreløpig kan du tenke på en selector som navnet på en tagg.

Det finnes en stor mengde ulike egenskaper med tilhørende verdier. Å få oversikt over alle egenskaper og tilhørende verdier som finnes, er derfor en stor jobb. Vi skal i løpet av denne boka ta deg gjennom de mest vanlige av disse.

Forholdet mellom CSS og HTML

Forholdet mellom CSS og HTML er i utgangspunktet ganske enkelt. Det første vi må gjøre er å sørge for at HTML-dokumentet referer til CSS-dokumentet. Dette gjør vi, som tidligere nevnt, ved å lage en `<link>`-tagg i `<head>`-delen av HTML-fila:

```
<head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stilarkfil.css" />
</head>
```

I selve CSS-fila angir vi ganske enkelt en selector og forteller hvordan vi ønsker at den skal se ut. Til nå har vi brukt taggnavn som selectorer, og det har fungert fint. Vi kan f.eks. ha noen paragrafer i HTML-fila:

```
<p>Første avsnitt</p>
<p>Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

og så i CSS-fila fortelle at disse skal vises med grå skrift

```
p {
    color: gray;
}
```

Problemet med taggnavn som en selector er imidlertid at regelen da vil gjelde alle tagger av denne typen. Vi kan ikke si at en fargeforandring kun skal gjelde en bestemt paragraf på denne måten.

Forsøk selv

1. Åpne mal-filen i din kodeeditor og skriv inn de tre paragrafene i `<body>`-delen:

```
<p>Første avsnitt</p>
<p>Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

2. Lagre med navnet *formatering.html* i en mappe kalt *kapittel2*.
3. Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som setter paragrafen til grå

```
p {
    color: gray;
}
```

4. Lagre med navnet *stilarkfil.css*
5. Vis *formatering.html* i din nettleser
6. Som du ser, blir alle paragrafenes tekst grå



Attributtene `id` og `class`

Dette problemet har man løst ved å benytte de to attributtene `id` og `class` i HTML.

Dersom vi ønsker å henvise til ett bestemt element i nettsiden, kan vi gi dette elementet en `id`:

```
<p id="ingress">Første avsnitt</p>
<p>Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

Deretter kan vi henvise til dette ene elementet fra CSS ved å henvise til ID-en med et #-tegn foran:

```
#ingress {
    font-weight: bold;
}
```

Kun ett element i hver nettside kan ha en bestemt ID. Det er ingenting i veien for at flere ulike nettsider hver kan ha ett element med samme ID.

Dersom vi ønsker å si noe om en gruppe elementer, kan vi gi disse en klasse gjennom attributtet `class`:

```
<p id="ingress">Første avsnitt</p>
<p class="brodtekst">Andre avsnitt</p>
<p class="brodtekst">Tredje avsnitt</p>
```

I CSS henviser vi så til klassen ved å benytte klassenavnet med et punktum foran som selector:

```
.brodtekst {
    font-style: italic;
}
```

Det er ingenting i veien for at ulike typer elementer kan være knyttet til samme klasse.

Et element kan gjerne både ha en `id` og en `class`.

Forsøk selv

Sørg for at fila `formatering.html` fortsatt er åpen, og endre de tre paragrafene. Gi den første en id, og de to neste en klasse:

```
<p id="ingress">Første avsnitt</p>
<p class="brodtekst">Andre avsnitt</p>
<p class="brodtekst">Tredje avsnitt</p>
```

Sørg for at `stilarkfil.css` er åpent i din kodeeditor og legg til følgende selector med egenskap og verdi:

```
#ingress {
    font-weight: bold;
}
```

```
.brodtekst {
    font-style: italic;
}
```

Lagre HTML- og CSS-fil. Sjekk resultatet i nettleser.

Navngiving av klasser og ID-er

Når vi skal navngi klasser og ID-er, er det litt viktig at vi tenker på hvilke navn vi benytter.

De formelle reglene sier at et slikt navn må starte med understrekk, bindestrek, eller en bokstav (a-z). Påfølgende tegn kan være tall, bokstaver, understreker eller bindestreker. Navnet må også bestå av minst to tegn. Med disse reglene er f.eks. navnene `qxer`, `ww33` og `zq_up` gyldige.

Selv om det er mulig, fraråder vi å benytte de norske tegnene æ, ø og å i navngivningen. Dette vil kunne gi problemer i enkelte nettlesere.



I tillegg til de formelle reglene bør vi følge en uformell regel om at navnene vi gir skal være fornuftige og forklare hva elementet er. Bruker vi navn som ikke gir mening, blir HTML- og CSS-koden svært vanskelig å lese både for deg selv og andre.

Ettersom mellomrom som en del av filnavnet ikke er lov, må vi finne en løsning på hvordan vi skal sette sammen ord. Standarden gir ingen klare føringer for dette, så her har vi minst tre mulige løsninger, der utviklere er uenige om hvilken som er best:

Bindestrek: informasjonsboks-med-annonce

Understrek: *informasjonsboks_med_annonse*

camelCase: *informasjonsboksMedAnnonse*

Vi kommer til å bruke camelCase-metoden i denne boka, men du står helt fritt til å velge noe annet.



CamelCase er vanlig å benytte for navngivning i programmering. Systemet består i at første ord begynner med liten bokstav, det som skulle vært påfølgende ord starter med stor.

Unngå også navn som forteller noe om utseendet. Dette kan endre seg over tid, og det blir veldig rart om et navn som indikerer ett utseende gir et annet. Gi heller elementene navn som forteller noe om hva de er. Følgende navn er derfor ikke å anbefale:

graaBoks

storeBokstaver

Disse er eksempler på bedre navn:

annonseboks

ingress

annonssetittel

Multiple class

Det er ingenting i veien for at et element kan følge flere klasser samtidig. Dette gjøres ved å ramse opp klassene adskilt med mellomrom i elementets classattributt:

```
<p class="infoboks uthevet viktig">Tekst</p>
```

Det er langt bedre å benytte flere klasser enn å ha kjempestore og spesifikke klasser. Ved å bryte opp designet i flere ulike klassedefinisjoner kan man gjenbruke deler:

```
<p class="infoboks uthevet viktig">Tekst 1</p>
```

```
<p class="infoboks uthevet">Tekst 2</p>
```

CSS-definisjonen kunne typisk vært:

```
.infoboks {  
    border-style: solid;  
    background-color: gray;  
}  
  
.uthevet {  
    font-weight: bolder;  
}  
  
.viktig {  
    text-decoration: underline;  
}
```



Hadde vi laget alt som en stor klasse, måtte vi laget en kopi for hver gang vi ønsker gjøre noe nesten likt, med mye overlapp i hver av kopiene:

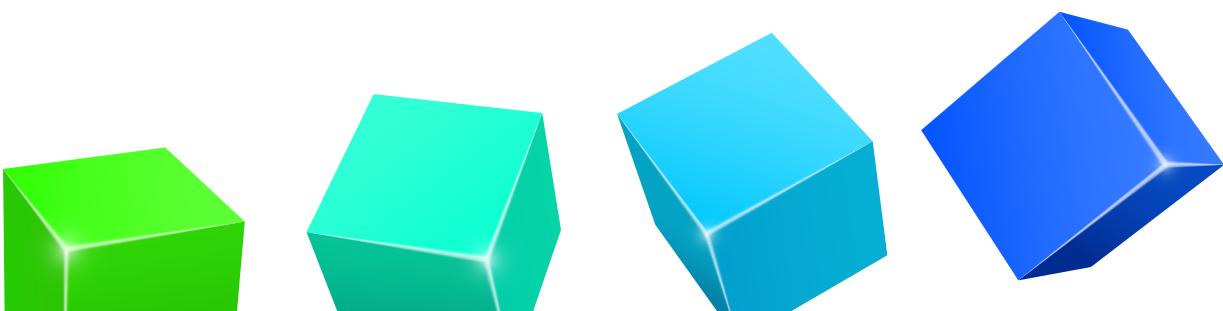
```
<p class="viktigUthevetInfoboks">Tekst 1</p>  
<p class="uthevetInfoboks">Tekst 2</p>
```

Her ville da CSS-definisjonen måtte være slik, og som du ser gjentas egenskapene.

```
.viktigUthevetInfoboks {  
    border-style: solid;  
    background-color: gray;  
    font-weight: bolder;  
    text-decoration: underline;  
}  
  
.uthevetInfoboks {  
    border-style: solid;  
    background-color: gray;  
    font-weight: bolder;  
}
```

Dette gjør at vi får flere steder å endre, dersom vi f.eks. vil endre bakgrunnsfarge på alle varianter av infoboks.

Dersom flere av klassene du bruker spesifiserer samme egenskap, skulle man tro at det var rekkefølgen de stod oppført i **class**-atributtet som angav hvem som «vant». Det er imidlertid slik at det er den klassen som står oppført/definert sist i CSS-dokumentet, som vil gjelde.





Forsøk selv:

- Åpne mal-filen og endre tekst i `<body>`-del som vist:


```
<p class="infoboks uthevet viktig">Tekst 1</p>
<p class="infoboks uthevet">Tekst 2</p>
```
- Endre koblingen til stilarket slik at den referer til en fil kalt `formatering2stil.css`.
- Lagre som `formatering2.html` i mappen `kapittel2`.
- Opprett en ny fil som du lagrer med navn `formatering2stil.css` og som har følgende innhold:

```
.infoboks {
    border-style: solid;
    background-color: gray;
}

.uthevet {
    font-weight: bolder;
}

.viktig {
    text-decoration: underline;
}
```

- Åpne `formatering2.html` i nettleseren. Hva er resultatet?

Taggene `` og `<div>`

Vi vil etterhvert oppdage at det ikke alltid er noen fornuftige elementer i HTML-koden som vi kan knytte et ønsket design til. Dette kan være fordi vi ønsker å gruppere sammen flere elementer, eller fordi vi kun ønsker å gjøre noe med deler av innholdet i et element.

HTML har to tagger med navnene `` og `<div>` som kan benyttes for å løse dette. Disse taggene har ingen betydning for nettleseren, og har heller ikke noen standardstil knyttet til seg. Det er først når vi selv beskriver utseendet til disse taggene i CSS at innholdet endres.

Taggen `<div>` benytter vi for å lage en blokk, og kan gjerne ha andre elementer inni seg. For eksempel kan den benyttes for å gruppere sammen paragrafer:

```
<div>
  <p>Første avsnitt</p>
  <p>Andre avsnitt</p>
  <p>Tredje avsnitt</p>
</div>
```

Dersom vi ikke ønsker at CSS-koden vi skriver skal gjelde alle `<div>`-tagger, må vi knytte en `id` eller en `class` til den. Deretter kan vi henvise til en `id` eller `class` fra CSS på vanlig måte.

```
<div id="brodtekst">
  <p>Første avsnitt</p>
  <p>Andre avsnitt</p>
  <p>Tredje avsnitt</p>
</div>
```

TIPS

Tagger av typen `<div>` benyttes også ofte til å markere større deler av nettsiden slik som hovedinnhold, bunntekst, meny osv. I HTML5 er det imidlertid egne tagger for dette som vi kommer tilbake til i neste kapittel.

Taggen med navn `` benyttes som regel for å markerer deler av innholdet i et element, slik som eksempelvis deler av en tekst i en paragraf. Også her bør vi sette en id eller en class for å senere kunne si noe i CSS om dette elementet.

`<p>Ole fikk 5 poeng og Per fikk 10 poeng.</p>`

Forsøk selv:

- Åpne mal-filen og endre tekst i `<body>`-del som vist:

```
<div>
  <h3>Katter</h3>
  <p>En flott katterase er bengalkatten, <span class="uthev">men pass på fordi den kan oppføre seg nokså vilt!</span></p>
</div>
<div>
  <h3>Hunder</h3>
  <p>Den irske setteren har en flott rød pels</p>
</div>
```



- Endre koblingen til stilarket slik at den referer til en fil kalt `divogspan.css`.
- Lagre som `divogspan.html` i mappen `kapittel2`
- Opprett en ny fil som du lagrer med navne `divogspan.css` og som har følgende innhold:

```
div {
  display: inline-block;
  background-color: linen;
  border: solid 2px black;
  width: 250px;
  padding: 20px;
  margin: 5px;
}
.uthev {
  color: orangered;
  background-color: yellow;
  font-weight: bold;
}
```

5. Åpne *divogspan.html* i nettleseren:

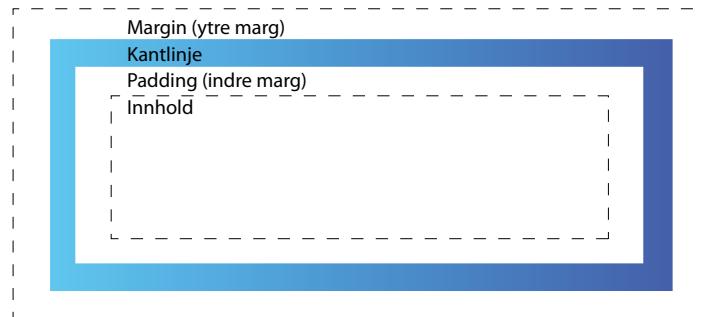


Vi angir **display: inline-block** for at div-ene skal stå etter hverandre - i tillegg til at vi kan oppgi bredde (og eventuelt høyde) på dem.

CSS Box model

Et viktig prinsipp i CSS er den såkalte *box model*. Den innebærer at alle elementer i en nettside ansees å være en firkantet boks med et innhold og en mulig kantlinje. Ved å gjøre denne forenklingen kan man lage et sett med egenskaper som gjelder alle typer elementer, i stedet for å lage et sett med egenskaper til hver type element.

Vi kan derfor vite at uansett type element, har vi mulighet til å legge på en kantlinje gjennom egenskapene **border-style** og **border-width**, vi kan sette en *ytre marg* gjennom egenskapen **margin**, og en *indre marg* gjennom egenskapen **padding**.



Vi kan f.eks. sette en kantlinje rundt en paragraf, og ha 2 piksler avstand mellom innholdet og kantlinjen, og 10 piksler avstand til omkringliggende elementer på følgende måte:

```
p {
    border-style: solid;
    border-width: 1px;
    padding: 2px;
    margin: 10px;
}
```

Mange synes det er vanskelig å skille **margin** og **padding** fra hverandre, men tenk på det så enkelt som at **padding** er avstand mellom selve innholdet og en eventuell kantlinje, og **margin** er avstanden mellom kantlinjen og et annet element.

TIPS

Det er viktig å ha et bevisst forhold til hva som skal være marg på utsiden av kantlinjen, og hva som skal være marg på innsiden av kantlinjen, også når elementer ikke har noen kantlinje på nåværende tidspunkt. Det er ikke utenkelig at du senere skal legge til en kantlinje, og da er det mye jobb å endre.

MERK

Vi kan alltid sette en størrelse på en box gjennom egenskapene **width** og **height**. Husk da imidlertid på at vi som standard justerer størrelsen på selve innholdet, og ikke omrisset utenfor ytre marg.

```
p {
    border-style: solid;
    border-width: 2px;
    padding: 2px;
    margin: 10px;
    width: 300px;
    height: 40px;
}
```

Vi kommer tilbake med en grundigere forklaring av alle disse og flere CSS-egenskaper tilhørende box model i kapittel 4.

Forsøk selv

- Åpne mal-filen og skriv inn en følgende i <body>-delen:

```
<h1>IT-1</h1>
<p>Faget IT-1 er et variert og artig fag</p>
```



- Endre linken til stilark slik at den referer til *boksstil.css*:

```
<link rel="stylesheet" type="text/css" href="stilark3.css" />
```

- Lagre med navnet *boks.html* i mappen *kapittel2*

- Opprett en ny fil som du gir navnet *boksstil.css* og lagrer i samme mappe

- Skriv inn følgende i stilarket ditt

```
p {
    border-style: solid;
    border-width: 2px;
    padding: 2px;
    margin: 10px;
    width: 300px;
    height: 40px;
}
```

- Vis i nettleser. Test ut ulike verdier for egenskapene **width**, **height**, **padding**, **margin** og **border-width**

Ulike nivåer av CSS

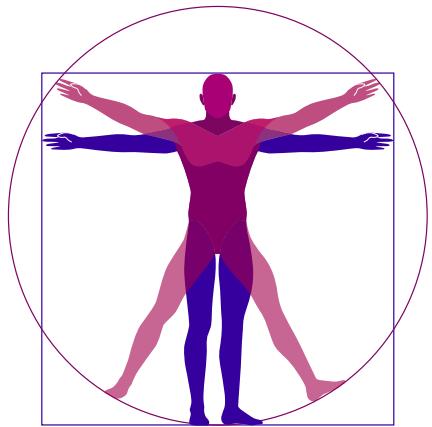
Når vi skal skrive CSS-regler for nettsiden, finnes det tre ulike *nivåer* å skrive koden på. Vi har alt sett på ett, som kalles *external* (eksternt). Her skriver vi CSS-koden i et eget dokument/fil, som så kobles inn gjennom en `<link>`-tagg i `<head>`-seksjonen av HTML-fila

```
<link rel="stylesheet" type="text/css" href="stilarkfil.css" />
```

Det er også mulig å legge inn CSS-kode direkte i HTML-fila gjennom såkalt internal (internt) CSS. I stedet for en `<link>`-tagg legger vi enn en `<style>`-blokk i `<head>`.

```
<head>
  <meta charset="utf-8" />
  <title>Vår første nettside</title>
  <style>
    body {
      background-color: lightgray;
      font-family: Arial;
    }

    h1 {
      text-decoration: underline;
    }
  </style>
</head>
```



Umiddelbart kan det virke langt mer praktisk å benytte teknikken med et internt stilark. Vi slipper jo da styret med to ulike filer. Husk da på at vi også mister muligheten til å benytte samme stilark på flere sider og må heller kopiere med oss koden. Kopiert kode er et mareritt når vi senere skal endre noe i utseendet.

Den siste metoden er såkalt inline CSS. Her legger vi til et attributt kalt `style` på enkelttagger, og sette reglene direkte.

```
<body style="background-color: lightgray; font-family: Arial;">
  <h1 style="text-decoration: underline;">Velkommen til vår ↗
    første nettside</h1>
  <p>Mer innhold vil komme etter hvert.</p>
  <p>Du kan også gå til <a href="side2.html">neste side</a>.</p>
</body>
```

Med inline CSS er det svært lett å fortelle hvilke tagger koden gjelder, ettersom man slipper en selector. Ulempen er at CSS og HTML blir veldig flettet sammen, og det er vanskelig å få en oversikt over «utseendet». Det er også mye jobb å endre utseendet, ettersom det ofte blir mange steder i HTML-fila som må endres.

Som du kanskje har merket, er vi noe «lunkne» i entusiasmen over internal og inline. Begge disse teknikkene blander sammen innhold og utseende mer enn nødvendig, samtidig som de gjør det mer uoversiktig ved store og kompliserte webområder.

Derfor anbefaler vi å benytte eksternt stilark.

Så hvorfor nevner vi de da? Jo, fordi i enkelte tilfeller kan de være nyttige. Skal vi f.eks. lage kun én nettside, og er sikker på at det aldri kommer flere som skal benytte samme design, kan internal være nyttig. Skal vi ha en nettside med svært lite CSS, kan inline benyttes.

Cascading

Det er heller ikke slik at vi nødvendigvis må velge enten eller. Det går an å benytte flere metoder samtidig. Vi kan til og med beskrive de samme reglene på alle tre nivåene samtidig.



Tenk deg at vi har et eksternt stilark som bl.a. forteller at alle paragrafer skal ha blå tekst og være understrekket. Dette stilarket er så knyttet til mange HTML-sider:

```
p {
    color: blue;
    text-decoration: underline;
}
```

På én av nettsidene som har dette stilarket kan vi imidlertid «overstyre» det at paragrafer blir blå, ved å inkludere et internt stilark som sier noe annet. Alt som står i et internt stilark «trumfer» det som står i et eksternt.

```
<style>
  p {
    color: red;
  }
</style>
```

Vi kan også «overstyre» dette interne stilarket på nettsiden ved å lage en inline-stil på en av paragrafene. Inline «trumfer» nemlig både internal og external.

```
<p>Første avsnitt</p>
<p style="color: green;">Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

Unntaket er imidlertid om en eksternregel har en mer spesifikk (detaljert) selector enn internregelen. Har du f.eks. definert en regel for paragrafer i internal, men forteller noe om en paragraf med en spesifikk ID i external, så vinner altså denne mer spesifikke beskrivelsen.

Dette prinsippet kan litt forenklet formuleres slik (med noen unntak):

«*Det er alltid den nærmeste eller mest spesifikke stilregelen som vinner*»

Dette er også det som ligger i C-en i CSS. Begrepet *cascading* kan oversettes med *sammenfallende* på norsk. Altså, man slår sammen alle ulike nivåer av stilark til ett felles stilark, og der hvor samme elementer har ulike beskrivelser benytter man prinsippet om at den mest spesifikke «vinner» for å avgjøre hva som skal velges.

Det går an å påvirke denne sammenslåingen. Det kan vi gjøre ved å legge til kodeordet **!important** bak en regel. Vi kan f.eks. skrive følgende i et eksternt stilark, og «vinne» over beskrivelser av paragrafer i både internal og inline:

```
p {
    color: blue !important;
}
```

Det er imidlertid anbefalt å ikke benytte **!important** i for stor grad, da det skaper veldig uoversiktig kode. Kan problemet løses ved å ha mer spesifikke regler i det eksterne stilarket, så gjør heller det.



Forsøk selv:

- Åpne mal-filen og skriv inn følgende tekst i **<body>**-delen:

```
<p>Første avsnitt</p>
<p>Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

- Endre linken til stilark slik at den referer til *cascadingstil.css*
- Lagre med navnet *side1.html* i mappen *kapittel2*
- Velg så å lagre denne på nytt med navnet *side2.html*. Du har nå to like HTML-filer
- Opprett stilarket *cascadingstil.css* som du kobler til begge filene:

```
p {
    color:blue;
    text-decoration: underline;
}
```

- Åpne begge sidene i nettleseren din, og kontroller at de vises likt
- Sørg for å stå i *side1.html*. Skriv inn følgende tekst i **<head>**-delen:

```
<style>
    p {
        color: red;
    }
</style>
```

- Lagre og oppdater visning av nettsiden i nettleseren. Nettsiden *side1.html* skal nå ha rød tekst i paragrafer. Det har ikke *side2.html*
- Stå fremdeles i *side1.html* i din kodeeditor og endre teksten i **<body>**-delen som vist:

```
<p>Første avsnitt</p>
<p style="color: green;">Andre avsnitt</p>
<p>Tredje avsnitt</p>
```

- Hva er nå forskjellen på designet i *side1.html* og *side2.html*? Hvilke stilregler trumfer høyest?

De to ekstra CSS-nivåene

Selv om de tre nivåene inline, internal og external er de som er vanligst å omtale, finnes det ytterligere to nivåer som er viktige å forstå. Dette er *user* og *browser* (også kalt *user agent*).

Alle nettlesere har definert et forholdsvis omfattende stilark for hvordan HTMLelementer skal stilsettes dersom ingenting annet blir sagt. Dette er grunnen til at en **<h1>**-tagg blir stor og med fet skrift, og en **<a>**-tagg blir blå og understrekket. Med andre ord et slags «standardstilark» for alle nettsider. Dette blir aktivt uten at vi refererer til det i HTML-koden vår.

Dette standardstilarket har laveste rang i CSS-nivåene, og alt vi gjør via inline, internal og external «trumfer» standardstilarkets regler. Disse reglene er med andre ord kun aktive dersom ingenting annet blir sagt.

Du kan finne din nettlesers standardstilark ved å gjøre et Google-søk etter «[nettlesernavn] user agent stylesheet». Da denne boka ble skrevet kunne man f.eks. Se Chrome (Webkit) sitt her:
<http://trac.webkit.org/browser/trunk/Source/WebCore/css/html.css>

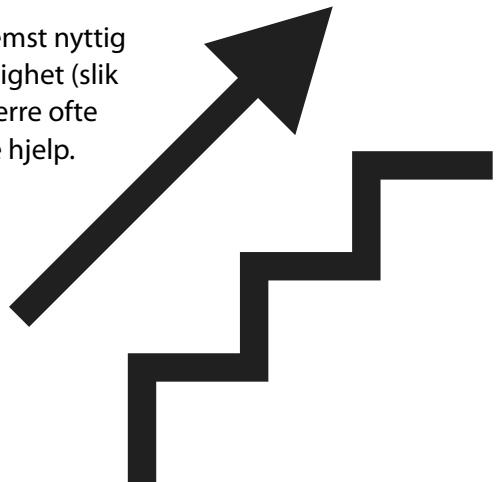
TIPS

Selv om det hadde vært ideelt, er man ennå ikke helt i mål med å koordinere alle nettleseres standardstilark. Derfor er de dessverre ikke helt identiske. Skal vi være helt sikre på at siden vil se ut slik vi ønsker, må vi derfor overstyre alle regler i et eget stilark vi selv lager.

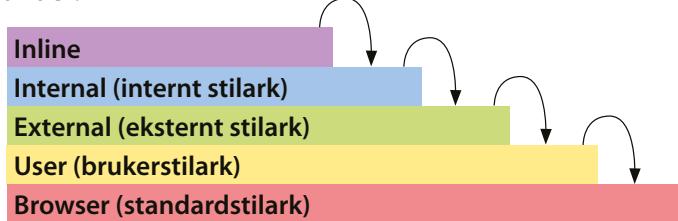
MERK

På nivået over standardstilarket ligger noe som kalles brukerstilarket. Dette er egne modifikasjoner vi som brukere kan sette og som overstyrer standardstilarket i nettleseren. I noen nettlesere gjøres dette via egne filer i mappen med innstillinger for nettleseren, i andre gjennom dialogbokser i selve nettleseren.

Dette nivået er ikke mye i bruk, og er kanskje først og fremst nyttig for de som trenger justeringer med hensyn på tilgjengelighet (slik som kontrast, skriftstørrelse osv). Disse overstyrer dessverre ofte av nettsidenes egne stilark, så det er ikke alltid til så mye hjelp. Enkelte nettlesere har imidlertid en innstilling der man kan sette brukerstilarket til det viktigste, og det som dermed «trumfer» alt. I noen nettlesere vil også en **!important** i brukerstilarket overstyre de andre stilarkene.



Totalt sett ser nivåinndelingen til stilark slik ut, der høyere nivåer trumfer nivåene under:



Validering av nettsider

Selv om de fleste nettlesere i dag er svært medgjørlige når det gjelder feil og mangler i HTML-koden, er det likevel veldig viktig at koden ikke inneholder feil.



Feil i koden kan medføre at nettsiden din vises helt uventet eller kanskje ikke i det hele tatt i enkelte nettlesere og enkelte typer systemer. Dette fordi nettleserne må gjette hva du mente, og det finnes ingen offisielle retningslinjer for hvordan disse gjettningene skal foregå.



Til nå har det vært en slags konkurranse blant nettleserne om å tolerere mest mulig feil. Brukerne har sett på nettsteder som ikke klarer å vise en nettside, som «dårlige», selv om det egentlig er de som har laget nettsiden som har gjort feil. I dag er det derimot langt mer fokus på hurtig og strømlinjeformet lasting av nettsider, så i tiden fremover vil nok mye av feiltoleransen forsvinne.



Det er også svært viktig at nettsidene er feilfri for at søkermotorene skal kunne tolke dem og ta dem med i sin oversikt over kjente nettsider på riktig måte.

I stedet for å studere koden vår i detalj selv så finnes det en god del ferdige verktøy som kan sjekke koden vår for feil. Standardiseringsorganet W3C laget et slikt verktøy for HTML-kode. Du finner validatoren på nettadressen:

<http://validator.w3.org>

This validator checks the markup validity of Web documents in HTML, XHTML, SIME, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CED documents, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our [open-source validator](#).

The W3C validators are developed with assistance from the Mozilla Foundation, and supported by community donations. [Donate](#) and help us build better tools for a better web.

[Home](#) [About](#) [News](#) [Docs](#) [Help & FAQ](#) [Feedback](#) [Contributors](#)

Copyright © 2016-2017 W3C. This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#). This service runs the W3C Markup Validator, [plädi](#).

Her har du tre ulike måter å validere koden din på:

- **Validate by URL:** Ved denne typen validering må man allerede ha lagt ut (nettsiden sin på en eller annen offentlig webserver. Man kan deretter skrive inn URL-en (nettadressen) til sin nettside og så få kontrollert denne.
- **Validate by File Upload:** Her kan man laste opp en HTML-fil man utvikler, og som ennå ikke er lagt på en webserver.
- **Validate by Direct Input:** Her kan man lime inn HTML-kode gjennom klipp og lim. Vær oppmerksom på at man vil få feilmeldinger på at det ikke er et gyldig dokument dersom man ikke limer inn hele HTML-koden.

Et alternativ til W3C sin offisielle validator er den du finner på
<http://validator.nu>

Tilsvarende finnes det et valideringsverktøy for CSS på nettadressen
<https://jigsaw.w3.org/css-validator/validator.html.en>

Dette verktøyet fungerer på samme måte som valideringsverktøyet for HTML.

Det er viktig å validere HTML og CSS koden jevnlig mens du utvikler nettsiden. Å validere til slutt medfører ofte såpass mange feil og et krav om så mange endringer at det nesten føles som å måtte begynne på nytt.

Kompatibilitet

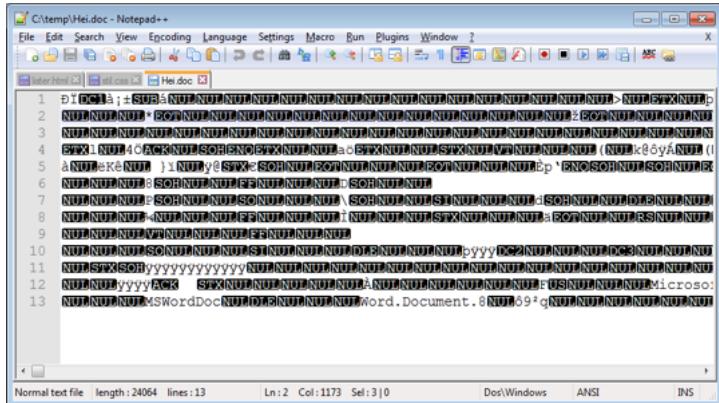
Selv om både HTML og CSS er godt spesifiserte standarder, er det dessverre ikke slik at alle nettlesere håndterer koden helt likt. Enkelte kodeord fungerer bare i noen nettlesere, og enkelte elementer får noe ulikt utseende.

I dag er dette et langt mindre problem enn tidligere, men det er fortsatt noe vi må ta hensyn til. Det er svært lurt å kontrollere nettsidene i flere nettlesere både underveis i utviklingen og før de legges ut på nettet.

Du må også etterstrebe å unngå å benytte kode som kun vil virke i bestemte nettlesere. Tiden der man kunne skrive at «denne nettsiden sees best i Internet Explorer versjon 7 og 8» er helt klart forbi. Brukere forventer å få servert like fine og funksjonelle sider i enhver nettleser og enhet.

En av fordelene ved HTML er at det er ren informasjon (tekst) som får tilleggsbetydning gjennom tagger. Selv uten en nettleser og uten forståelse for HTML-koder kan vi åpne dokumentet i en editor og plukke ut innholdet. Vi mister informasjon slik som at det er en overskrift, men teksten i seg selv får vi tak i.

Forsøker du å derimot å åpne en Word-fil i en teksteditor, vil du ikke kunne hente ut informasjonen like enkelt selv om den faktisk er der.



Dette prinsippet er også svært viktig for såkalt *kompatibilitet* mot nettlesere. HTML utvikles stadig, og nye tagger og attributter dukker opp, mens andre forsvinner. Det er ikke slik at alle nettlesere støtter alle mulighetene som finnes.

De som lager nettleserne, kunne gått for en strategi der de viste feilmeldingen «Denne nettsiden kan ikke vises» med en gang en ukjent tagg eller attributt dukket opp. I stedet har de valgt prinsippet om at dersom en tagg kun legger til betydning på tekst, og om taggen ikke er kjent, så vises bare det mellom start-taggene og slutttaggen uten noen endringer. Opplevelsen av nettsiden blir dårligere, men den fungerer.

På samme måte som en nettleser «hopper over» alle tagger den ikke forstår noe av, gjør nettleseren det samme med CSS. Alle selectorer, egenskaper eller verdier som er ugyldige for en nettleser, vil resultere i at endringen de indikerer, ikke blir utført. Resten av stilarket som er kjent blir derimot utført som det skal.

Relative og absolutte referanser

En stor del av HTML-koding består i å referere til andre ressurser/filer. Det kan f.eks. være som **src** i en ****-tagg eller **href** i en **<a>**-tagg.

Foreløpig har vi kun referert til ressurser som er plassert samme sted (samme mappe) som HTML-fila selv. Vi benytter da en såkalt relativ referanse ved at vi kun opplyser filnavnet, og nettleseren forstå at det er i forhold til der nettsiden er plassert.

Det andre ytterpunktet er såkalte absolute referanser. Her benytter vi en helt utvetydig angivelse av hvor ressursen er. På nett er dette det vi kjenner som en URL, og som starter med *http://* eller *https://*. Vi benytter URL for å koble til et element som ikke er plassert på samme sted som HTML-fila. Vi kan f.eks. lage en link fra vår nettside til Gyldendal sin nettside på følgende måte:

Besøk Gyldendals nettsider

Det er viktig å ha med `http://` eller `https://` først i en absolutt referanse. Vi er vant til at nettleseren forstå snarveier slik som `vg.no` eller `www.vg.no`, men i HTML må de være skrevet fullt ut.

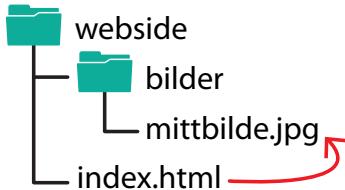


Til nå har vi kun nevnt relative referanser til filer på samme plassering som HTML-fila. Vi kan imidlertid også benytte referanser som angir plassering i andre mapper relativt til mappen HTML-fila er plassert i.

Har du f.eks. en egen undermappe kalt bilder, kan vi referere til fila på følgende måte:

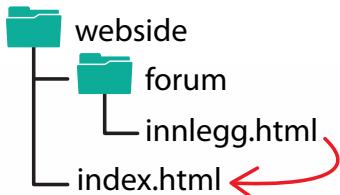
```

```



Vi kan også referere fra en undermappe og «oppover» i filstrukturen. Hvis vi i fila `innlegg.html` i mappa forum ønsker å ha en link tilbake til hovedsiden `index.html`, kan vi gjøre det ved hjelp av den spesielle skrivemåten `.. /` som betyr «en mappe opp»:

```
<a href=".. /index.html">Tilbake til hovedsiden</a>
```



Når skal man benytte hvilken metode?

Logisk nok må du alltid ha absolutte referanser når du refererer til ressurser som ligger andre steder (andre domener) enn der nettsiden er plassert.

Referanser internt på en nettside kan imidlertid være både absolutte og relative.

Referansen

```
<a href="http://www.mittdomenenavn.no/index.html">Tilbake til hovedsiden</a>
```

fungerer like godt som

```
<a href="../index.html">Tilbake til hovedsiden</a>
```

Fordelen med å benytte relative referanser internt på en nettside er at hele webområdet lett kan flyttes til en annen plassering uten å måtte skrive om alle referansene.

Kommentarer

Når vi skriver HTML- eller CSS-kode, bør vi være flinke til å kommentere koden. En kommentar er tekst som ikke nettleseren bryr seg om og som den som besøker nettsiden, ikke ser.



Kommentarer er til for å nærmest lage små «huskelapper» til deg selv i koden.

Foreløpig har ikke behovet for kommentarer vært det største, men når kildekoden bak nettsiden blir omfattende og komplisert kan det være veldig kjekt. Dersom flere skal samarbeide om utvikling av nettsider er det også en god ide å kommentere i koden slik at andre lettere kan overta andres arbeide.

Kommentarer settes inn i HTML ved hjelp av `<!-- -->`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Vår første webside</title>
    <!-- Her kobler vi inn stilarket stil.css -->
    <link rel="stylesheet" type="text/css" href="stil.css" />
  </head>

  <body>
    <h1>Velkommen til vår første nettside</h1>
    <p>Mer innhold vil komme etter hvert.</p>
    <!-- Her lager vi en link til side2.html -->
    <p>Du kan også gå til <a href="side2.html">neste side</a>. </p>
  </body>
</html>
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Tilsvarende kan vi sette inn kommentarer i CSS ved hjelp av tegnsekvensene /* og */.

```
body {  
    /* Setter bakgrunnsfargen på nettsiden til lys grå */  
    background-color: lightgray;  
    /* Setter fonten til Arial, med Helvetica Neue, Helvetica og sans-serif  
       som mer generelle backup-fonter dersom Arial ikke er installert hos  
       brukeren */  
    font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;  
}  
  
h1 {  
    /* Lager en understrek på alle overskrifter */  
    text-decoration: underline;  
}
```

Kommentarer virker alltid meningsløse på små og enkle prosjekter. Det er imidlertid veldig lurt å lære seg til å kommentere små prosjekter, så man har vanen på plass etter hvert som prosjektene vokser i størrelse.

Husk at kommentarene er tilgjengelige for de brukerne som velger å vise kildekoden bak en publisert nettside. Unngå å skrive kommentarer du ikke vil at andre skal lese.



Kommentarer kan imidlertid også benyttes til å midlertidig fjerne kode. Hvis ikke nettsiden vises som du har tenkt, kan du forsøke fjerne deler av koden bak (HTML / CSS), og se om du kan finne feilen på den måten.

Alle ordinære HTML-tagger kan enkelt «kommenteres bort». Vi kan f.eks. enkelt «skru av» stilarket midlertidig, uten å fysisk fjerne koden:

```
<head>  
    <meta charset="utf-8" />  
    <title>Vår første webside</title>  
    <!--link rel="stylesheet" type="text/css" href="stil.css" /-->  
</head>
```



Tilsvarende kan vi midlertidig fjerne egenskaper eller regler fra CSS, for å se hvordan utseendet blir uten:

```
body {  
    background-color: lightgray;  
    /* font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif; */  
}  
  
/*h1 {  
    text-decoration: underline;  
}*/
```

En siste kjekk effekt av kommentarer i HTML er når vi får mange lignende seksjoner/blokker inni hverandre. Typisk skjer dette ved bruk av <div> på mer omfattende nettsider. Da kan vi sette en kommentar bak hver <div>-tagg for å indikere hvilken ID eller klasse den avslutter.

```
<div id="wrapperr">  
    ...  
    <div id="boks1">  
        ...  
        <div id="content">  
            ...  
        </div><!-- Slutt content -->  
        ...  
    </div><!-- Slutt boks1 -->  
    ...  
</div><!-- Slutt wrapperr --->
```

Mer informasjon om HTML og CSS

Det er ikke mulig å bli fullt utlært på HTML og CSS ved å lese en bok. Vi kan gi deg en god forståelse for prinsippene og en oversikt over de vanligste kodene. Etter hvert som du begynner å lage mer avanserte nettsider, er du imidlertid avhengig av flere ressurser.

Vi vil derfor gi deg tips om noen ressurser som kan være nyttige for å finne mer informasjon om HTML og CSS.

W3C og WHATWG

W3C (World Wide Web Consortium) er standardiseringsorganet for HTML og CSS. HTML5-standarden utarbeides også av WHATWG (Web Hypertext Application Technology Working Group). Det er altså disse gruppene som styrer utviklingen av språkene. På deres nettsider finner du tekniske spesifikasjoner av språkene. Dette kan være noe detaljert for de fleste, men kjekt å slå opp i om man lurer på noe helt spesielt.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

WHATWG sin Living standard for HTML5:

<https://html.spec.whatwg.org/multipage/>

W3C sin Working Draft for HTML5:

<http://www.w3.org/TR/html5/>

W3C sin beskrivelse av CSS-nivåene:

<http://www.w3.org/TR/CSS/>

W3Schools

W3Schools baserer seg på å tilnærme seg HTML og CSS ved hjelp av guider (tutorials) og kjørbare eksempler. Dette er kanskje det beste stedet for en nybegynner å starte: <http://www.w3schools.com>.

Google

Som med alt annet, er Google også en stor ressurs i forbindelse med HTML og CSS. Her kan du f.eks. søke etter «taggnavn HTML», «egenskapsnavn CSS» eller «HTML and CSS tutorial» og finne masse ressurser.

StackOverflow

Nettstedet stackoverflow.com er et stort forum der brukere spør om hjelp til konkrete problemer og medlemmer svarer. Brukere kan så stemme frem de beste svarene. Forumet benyttes til all slags teknisk utvikling, også HTML og CSS. Selv om nettstedet har en egen søkemotor, er nok det å finne stackoverflow-treff blant Google-søk den enkleste måten å bruke forumet på.

Andre nettsiders kode

En fordel med nettsider er at du også enkelt kan få frem kodene som andre har benyttet for å lage sine nettsider, for så å lære av disse. Når du besøker en nettside du lurer på hvordan er laget, trykker du **Ctrl + U** eller velger *Vis kildekode/Vis sidekilde* fra menyen i nettleseren.



3 Strukturere websider

I dette kapitlet vil du lære

- om formatering av tekst
- om strukturelle tagger
- om bilder
- om linker
- om tabeller
- om lister
- om lyd og video

I dette kapittelet vil vi ta et dypdykk i en rekke ulike tagger og tilhørende attributter som finnes i HTML. De aller fleste HTML-taggene er til for å strukturere innholdet. Det vil si at vi forteller hva de ulike delene av innholdet (teksten) betyr og hvordan det hører sammen med resten av innholdet.

Å strukturere innholdet kan derfor være alt fra å angi at det er en paragraf eller overskrift, til å sette innholdet inn i en tabell eller liste. Tilsvarende vil bilder og linker være en del av innholdet og strukturen på nettsiden.

Formatering av tekst

Som du har sett tidligere, benyttet vi taggen `<h1>` for å formtere overskrifter i fet skrift og med en stor skriftstørrelse. Det er lett å tro at denne formateringen er en hurtigfunksjon for å endre utseende på teksten, i stedet for å benytte CSS, men det er ikke riktig.

Når vi benytter *formatteringstagger* gir vi først og fremst teksten en *betydning* i sammenhengen. Markering av teksten med for eksempel heading 1, forteller at dette er hovedoverskriften på nettsiden - altså den viktigste overskriften. Dette vil for eksempel søkemotorer benytte for å forstå hva nettsiden handler om. En ekstra effekt av formatteringstaggene er at nettsideleserne har et standard stilark som setter utseendet på teksten, altså det som de fleste tror er hensikten med taggene. Dette gjør det for at det skal være lettere for brukeren å se hva som er f.eks. en overskrift, dersom ikke noe annet utseende er angitt.

Nedenfor følger en kort forklaring til de vanligste tekstformatene:

Paragrafer

Taggen `<p>` lager det vi i ordinær tekst ville kalt en paragraf eller avsnitt. Hver blokk med tekst i denne boka ville typisk vært en paragraf-tagg. Som standard blir det satt inn litt avstand før og etter avsnittet.

`<p>Dette er første avsnitt, og teksten vil bli vist som en enkelt blokk</p>`
`<p>Dette er andre avsnitt, og teksten vil bli vist som en enkelt blokk</p>`

Dette er første avsnitt, og teksten vil bli vist som en enkelt blokk

Dette er andre avsnitt, og teksten vil bli vist som en enkelt blokk

En paragraf vil automatisk få linjebrudd i seg dersom teksten blir bredere enn vinduet nettsiden vises i. Du kan også selv sette inn linjebrudd i en paragraf ved hjelp av taggen `
`. Merk deg at linjebrudd er en tagg som starter og stopper «i seg selv», altså at den kun markerer punktet det linjebruddet skal stå.

`<p>Dette er første avsnitt,
 og teksten vil bli vist som en enkelt blokk</p>`
`<p>Dette er andre avsnitt, og teksten vil bli vist som en enkelt blokk</p>`

Dette er første avsnitt,
og teksten vil bli vist som en enkelt blokk

Dette er andre avsnitt, og teksten vil bli vist som en enkelt blokk

Det er også mulig å markere punkter der man ønsker at linjebrudd skal settes, dersom det skulle bli behov for linjebrudd. Dette gjøres ved hjelp at taggen `<wbr />`. Denne taggen benyttes altså som en `
`-tagg, men gir kun linjebrudd i ved behov.

Husk at du ikke skal lage «design» ved å sette flere `
` etter hverandre. Taggen `
` skal kun benyttes dersom linjeskift er en del av informasjonen. Mange bryter mot dette og lager avsnittskille ved å benytte `

` i stedet for flere `<p>`-tagger.





Forsøk selv:

1. Opprett en mappe med navnet *kapittel3* i mappen *webutvikling*
2. Åpne mal-filen
3. Skriv inn teksten under i **<body>**-delen:

```
<p>Dette er første avsnitt, <br /> og teksten vil bli vist som en enkelt blokk</p>
<p>Dette er andre avsnitt, og teksten vil bli vist som en enkelt blokk</p>
```

4. Lagre med navnet *paragrafer.html* og vis i nettleseren din

Overskrifter

Overskrifter finnes i ulike utgaver og er nummerert fra 1 som er den mest betydningsfulle overskriften i nettsiden og helt ned til 6, som er den minst betydningsfulle overskriften. Disse er da angitt med taggene **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** og **<h6>**.

```
<h1>Hovedoverskrift</h1>
<p>Tekst</p>
<h2>Underoverskrift</h2>
<p>Tekst</p>
<h2>Underoverskrift</h2>
<p>Tekst</p>
```

Hovedoverskrift

Tekst

Underoverskrift

Tekst

Underoverskrift

Tekst

Du bør forsøke å organisere innholdet i nettsiden med riktige overskrifter, og dermed lage en hierarkistruktur på informasjonen. Vi bør etterstrebe å lage en struktur der vi ikke går umiddelbart fra f.eks. nivå 2 til 5 eller at vi benytter nivå 3 som hovedoverskrift.

TIPS

Verktøyet du finner på nettadressen <https://gsnedders.html5.org/outliner/> kan presentere «strukturen» i overskriftene på en nettside du angir. Mange store kjente nettsider en dårlig struktur - test ut ulike nettsider via denne tjenesten.

Utheving og redigering

Taggene **** og **** benyttes for å utheve viktige deler av en tekst. Dette er spesielt nyttig for søkemotorer og skjermlesere, men vil også gi et visuelt tegn til leseren.

Kort fortalt kan vi si at **** markerer en viktig del av teksten. Det at noe er markert med **** forandrer imidlertid ikke mening på teksten. Taggen **** angir at det skal legges trykk på denne delen av teksten og gir annen mening til teksten (slik som «det var kun *tre* personer der»).

Det finnes også en tagg kalt **<mark>** som angir en relevant del av teksten for leseren i en viss kontekst, men en slik markering forandrer ikke meningen. Typisk er **<mark>** det man ville «gulete ut» i en bok når man leser til eksamen.

```
<p>Tekst med <em>em</em> i bruk.</p>
<p>Tekst med <strong>strong</strong> i bruk.</p>
<p>Tekst med <mark>mark</mark> i bruk.</p>
```

Tekst med *em* i bruk.

Tekst med **strong** i bruk.

Tekst med **mark** i bruk.

Vi kan også markere tekst som skal representeres som «fjernet» med taggen ****, tekst som er ugyldig/utdatert/feil med taggen **<s>** og ny tekst som er satt inn i forbindelse med en oppdatering med taggen **<ins>**.

```
<p>Tekst med <del>del</del> i bruk.</p>
<p>Tekst med <s>s</s> i bruk.</p>
<p>Tekst med <ins>ins</ins> i bruk.</p>
```

Tekst med ~~del~~ i bruk.

Tekst med ~~s~~ i bruk.

Tekst med ins i bruk.

Husk at dersom du kun er ute etter den grafiske effekten av *kursive*, gjennomstreking, **bold** og understreking skal dette gjøres med CSS, som vi vil lære mer om i neste kapittel.



Vi kan også lage såkalt *subscript* og *superscript* tekst. Dette forbinder kanskje flest med matematiske formler og kjemiske forbindelser, der enkelte tegn er hevet eller senket. Også fotnoter kan dra nytte av denne teknikken.

<p>Den kjemiske formelen for vann er: H₂O</p>
 <p>Formelen for masseenergi er: E=mc²</p>

Den kjemiske formelen for vann er: H₂O

Formelen for masseenergi er: E=mc²

Forhåndsformatert tekst

Taggen <pre> (forkortelse for preformatted) kjennetegnes ved at den ikke automatisk justeres i forhold til nettleservinduets størrelse, og at innholdet beholder formatering fra HTML-koden.

Dette er den eneste taggen der du i kildekoden kan benytte tabulator eller ordinære linjeskift, og så få dette vist i nettsiden. Også flere mellomrom på rad vil vises, i motsetning til at de vanligvis slås sammen til ett mellomrom.

Taggen er derfor velegnet til for eksempel å presentere programkode hvor du vil unngå feilaktige linjeskift. Den vises også med en *monospace-font* som gjør at alle tegnene er like brede, noe som f.eks. gjør den egnet for såkalt *ASCII-grafikk*.



Forsøk selv:

1. Lag en ny fil kalt *formatering.html* ut i fra mal-fila
2. Skriv inn koden under (i <body>-delen av dokumentet), lagre og forhåndsvis i nettleser

<p>Tekst med em i bruk.</p>
 <p>Tekst med strong i bruk.</p>
 <p>Tekst med <mark>mark</mark> i bruk.</p>

3. Legg til teksten under i dokumentet *formatering.html*, lagre og forhåndsvis i nettleser

<p>Tekst med del i bruk.</p>
 <p>Tekst med <s>s</s> i bruk.</p>
 <p>Tekst med <ins>ins</ins> i bruk.</p>

4. Legg til teksten under i dokumentet *formatering.html*, lagre og forhåndsvis i nettleser

<p>Den kjemiske formelen for vann er: H₂O</p>
 <p>Formelen for masseenergi er: E=mc²</p>

Tegnsett

Angivelsen av et tegnsett forteller nettleseren hvordan informasjonen i HTML-dokumentet skal presenteres. Noen tegn, slik som a-z, tall og spesialtegn slik som punktum, parenteser og utropstegn er felles for de fleste tegnsett. Resten av tegnene varierer, og den samme informasjonen kan presenteres ulikt dersom den tolkes med forskjellige tegnsett.

Du har kanskje allerede lagt merke til enkelte nettsider som ikke viser de norske tegnene riktig?

Det fikk, rste ferietipset er: "Vær forsiktig med å koble til ukjente trådløse nettsteder"

For oss nordmenn er det to ulike tegnsett som er aktuelle å benytte:

- **ISO-8859-1 (latin-1)** - Et eldre tegnsett som utvider ASCII-standarden med blant annet de norske tegnene.
- **utf-8 (unicode)** - Et nytt tegnsett som har som mål å være det eneste tegnsettet vi behøver, da det forsøker inneholde alle tenkelige tegn.

Det er altså utf-8 alle nye nettsider vi lager bør benytte. Selve tegnsettet angir vi som tidligere nevnt i taggen `<meta>` i `<head>`-delen av nettsiden:

```
<head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stilarkfil.css" />
</head>
```

Like viktig som å angi tegnsett i HTML-fila, er å sørge for at fila faktisk er lagret med dette tegnsettet som vi angir. Vi beskrev dette i kapittel 1 under lagring av filer. Selv om vi har valgt et tegnsett er det imidlertid noen tegn som ikke er tillatt å benytte som en del av teksten i et HTML-dokument. Dette er såkalte *meta-tegn* som er en del av HTML-språket. Bruker vi de som en del av teksten, blir nettleseren forvirret over hvordan de skal tolkes:

`<p>Tegnet < benyttes for å uttrykke mindre enn</p>`

Som du ser av dette eksempelet benyttes tegnet < i teksten, men nettleseren vil tro at vi forsøker starte på en tagg. Løsningen er å benytte en alternativ representasjon av tegnet. Tegnet < kan skrives som <; dersom vi ønsker den tekstlige betydningen:

`<p>Tegnet < benyttes for å uttrykke mindre enn</p>`

I nettsiden vil teksten vises som vi forventer

Tegnet < benyttes for å uttrykke mindre enn

Det er ikke mange slike tegn vi må erstatte, men følgende må vi passe på:

<	<
>	>
&	&

I attributtverdier (det som står mellom " og ") må vi også erstatte alle forekomster av " med ". Dette gjelder imidlertid kun attributtverdier, og ikke selve teksten på nettsiden.

I tillegg er det noen tegn som er kjekke å kjenne til. Det er f.eks. mye jobb å finne frem tegnet ©, da det ikke finnes på tastaturet. Dette kan i HTML enkelt skrives som ©

Du har kanskje også lagt merke til at i HTML blir flere påfølgende mellomrom erstattet med ett mellomrom under visning. Ønsker vi å faktisk ha flere mellomrom må disse representeres ved hjelp av



Husk at du ikke skal lage design på nettsiden ved hjelp av mellomrom. Da skal heller CSS benyttes. Benytt kun når mellomrom er en del av informasjonen.

Bilder

Å sette inn et bilde som en del av innholdet gjøres som vi har sett ved hjelp av -taggen. Taggen i seg selv har ikke så mange variasjoner:

```

```

Det kan virke rart å omtale bilder som en del av struktur og informasjon, men et bilde er informasjon på samme måte som tekst. Vi skal senere se at bilder også kan settes inn ved hjelp av CSS, og det naturlige skillet går på om bildet er en del av informasjonen eller designet. Mer om dette i kapittel 4.

Som tidligere nevnt er det de to attributtene **src** og **alt** som er vesentlige for en -tagg. Egenskapen **src** forteller hvor bildet befinner seg, og kan være en absolutt eller relativ referanse. Egenskapen **alt** inneholder en alternativ tekst som vil vises for de som ikke kan se bildet eller om noe gikk feil under lasting.

Mange programmer lagrer filendelsen med store bokstaver, slik som JPG. Pass på å få referansene riktig når det gjelder store/små bokstaver for å unngå overraskelser når nettsidene publiseres til en server som skiller på dette. Helst bør alle filnavn kun bestå av små bokstaver.



En nettside vil ofte bestå av mange bilder. For å få en bra organisering og gjøre det enklere å finne frem i filene et nettsted består av, kan det være veldig lurt å lage en egen mappe til bildene du kaller *bilder*, *images*, *img* eller lignende. Fra HTML-koden og -taggen refererer du da til denne mappa sammen med filnavnet: *bilder/mittbilde.jpg*



Bildeformater

På Internett brukes det i prinsippet tre ulike bildeformater. De kan se like ut, men er ganske så forskjellige. Ulikhetene ligger blant annet i antall farger som kan representeres, måten bildedataene blir lagret på, og størrelsen på selve bildet. Jo større filen er, desto lengre tid tar det for brukeren å laste ned bildet.

Det er viktig at du vet hva som er forskjellen på de ulike formatene, og når de bør benyttes. Under følger en kort oversikt:



JPG (JPEG)

Et JPG-bilde lagres vanligvis i 16,7 millioner farger. Formatet inneholder også en kraftfull komprimering spesielllaget for fotografiske bilder. JPG er så godt som standarden for fotografiske bilder på nettet og er det vanligste formatet på bilder tatt med digitalkameraer.



GIF

Et bilde som er lagret i GIF-formatet, kan maksimalt ha 256 farger. Det gjør at bildet blir mindre i filstørrelse og derfor lastes raskere. På grunn av de begrensede fargene egner GIF seg best til tegninger og grafikk, ikke fotografiske bilder. I tillegg kan et GIF-bilde være gjennomsiktig. Et GIF-bilde kan også være animert.



PNG

PNG, som uttales «ping», er et format som har som målsetting å erstatte GIF. Formatet er spesialtilpasset formålet med grafikk på Internett. PNG har bedre komprimering og fargegjengivelse enn GIF, men støtter i seg selv ikke animasjoner. Transparens (gjennomsiktighet) er imidlertid en del av PNG.



Å benytte andre formater enn disse tre kan fungere i enkelte nettlesere. Bl.a. støtter Internet Explorer i flere av versjonene formatet BMP (bitmap). Du må imidlertid unngå andre formater om du ønsker å være sikker på å få vist bildet i alle nettlesere. De fleste andre formater, slik som BMP, er også dårlig egnet for nettsider på grunn av mangelen på komprimering.

Størrelser

Når vi setter inn et bilde ved hjelp av ``-taggen så får bildet i nettsiden samme dimensjoner som selve bildefila har. Noen bilder vil derfor ta liten plass på nettsiden, andre mye.

Vi kan påvirke størrelsen til visningen av bildet gjennom attributtene `width` og `height` som er en del av ``-taggen:

```

```

Størrelsen på bildevisningen er imidlertid ofte en del av designet, og ikke informasjonen, så trenden er at stadig flere gjør dette ved hjelp av CSS. Bilder er som andre tagger et firkantet element i box model-prinsippet.

```


#bildeavmeg {
    width: 50px;
    height: 75px;
}
```



Angir vi kun `width` eller kun `height` vil nettleseren automatisk beregne den andre, og beholde proporsjonene til bildet. Dette gjelder både om vi gjør skaleringen med HTML eller CSS.

En ting du skal være klar over er at å endre visningsstørrelsen til bildet ikke påvirker selve bildefila som skal hentes ned. En vanlig feil er å sette inn et bilde tatt med et digitalkamera/mobilkamera og en oppløsning på kanskje 3552 x 2448 piksler. Dette kan f.eks. ha en filstørrelse på 15 MB. Deretter skalerer man bildet når man setter det inn i HTML-koden til 400 x 300 piksler.

Problemet nå er at selve bildefila fortsatt er på 15 MB, selv om man benytter bildet i en kvalitet som kanskje ikke hadde trengt en bildefil på mer enn 1 MB. Hele fila lastes ned til brukeren fra serveren, og dette tar unødvendig lang tid.

Når du utvikler nettsiden merker du ikke at det å benytte en «overdimensjonert» fil tar lang tid å laste. Dette fordi du arbeider lokalt, eller at bildefilen er cachet (mellanlagret) i nettleseren fra tidligere visninger.



Anbefalingen er derfor å alltid skalere bildene til en oppløsning som tilsvarer den oppløsningen du skal benytte i nettsiden. Du vil lære mer om hvordan du gjør dette i del tre av denne boka.

Ta alltid vare på originalfila ved nedskalering av bildefiler. Det er ikke mulig å skalere den opp igjen uten å miste kvalitet, om du senere skulle endre meningen om ønsket størrelse.



Figurer

Med HTML5 kom også de to nye taggene `<figure>` og `<figcaption>`. I sin enkle form benyttes taggene for å sette en bildetekst som skal være synlig for brukeren.

```
<figure>
    
    <figcaption>Bilde av giraffen vår</figcaption>
</figure>
```



Bilde av giraffen vår

Teksten vil alltid følge bildet, uansett plassering. Enda mer fornuftig blir **<figure>**-taggen dersom vi samler flere bilder i samme figur:

```
<figure>
    
    
    <figcaption>Bilde av giraffen vår og tvillingbroren hans</figcaption>
</figure>
```



Bilde av giraffen vår og tvillingbroren hans

Ved å samle flere bilder i samme **<figur>**-tagg kan vi enkelt flytte og behandle de som en gruppe. Teksten vi angir i **<figcaption>** vil da også gjelde hele gruppen.



Forsøk selv:

1. Åpne mal-filen og lagre denne med navnet *figur.html* i mappen *kapittel3*
2. Sørg for å lagre fila *froskliten.jpg* i mappen *kapittel3* – du finner fila i øvingsfilene til boka
3. Skriv inn teksten under i **<body>**-delen

```
<figure>
    
    
    <figcaption>Bilde av frosken vår og tvillingbroren hans</figcaption>
</figure>
```

4. Lagre og vis fila i nettleseren

Linker

En sentral del av nettsider er linker som knytter flere nettsider sammen. Som vi tidligere har sett så lager vi linker ved å sette inn en `<a>`-tagg rundt det som skal bli klikkbart. I tillegg må vi sette `href`-attributtet (hyperreferanse) for å referere til ressursen brukeren skal navigeres til.

```
<p>Gå til <a href="side2.html">neste side</a>.</p>
```

Det finnes også et attributt med navn `target` som vi kan benytte for å fortelle hvordan ressursen vi linker til skal åpnes. Standard verdi her er `_self`, som betyr at ressursen skal åpnes i samme fane. Vi kan imidlertid sette den til å være `_blank`, noe som betyr at en ny fane skal åpnes:

```
<p>Gå til <a href="side2.html" target="_blank">neste side</a>.</p>
```

De nettlesere som ikke støtter faner, vil åpne linker av typen `_blank` i nye nettleservinduer

Forsök å designe nettsider slik at du i minst mulig grad trenger å benytte `_blank`. Idéen med webben er at man skal navigere frem og tilbake, ikke åpne nye nettsider hele tiden.

Det er ikke gitt at det som skal bli klikkbart må være en tekst. Vi kan også gjøre om bl.a. bilder til linker ved å plassere ``-taggen inne i `<a>`-taggen:

```
<a href="side2.html">
  
</a>
```

Husk at vi ikke nødvendigvis trenger referere til en nettside i `href`-attributtet. Vi kan fint også referere til andre typer filer som da enten åpnes i nettleseren om den har innebygget visning (typisk bilder, PDF osv.) eller lastes ned.

```
<p>Se vår <a href="manual.pdf">brukermanual</a>.</p>
```

De fleste nettlesere i dag støtter også det å lage en link til en e-postadresse. Klikker brukeren på linken starter så brukerens standard e-postklient med e-postadressen ferdig utfylt. I `href`-attributtet må vi inkludere kodeordet `mailto`: før e-postadressen:

```
<p>Send oss gjerne en <a href="mailto:kontakt@domene.no">e-post</a>.</p>
```





Forsøk selv:

- Legg til teksten under i en fil du kaller *linker.html* basert på mal-filen

```
<p>Gå til <a href="figur.html">forrige side</a>.</p>
```

- Lagre og forhåndsvis i nettleser – hva skjer når du klikker på linken?
- Endre linken som vist under og test så i nettleser:

```
<p>Gå til <a href="figur.html" target="_blank">forrige side</a>.</p>
```

Hva skjer nå?

- Endre i koden din slik at bildet av en frosk blir en hyperkobling (bildet finner du i øvingsfilene til boka):

```
<a href="figur.html">
  
</a>
```

- Lagre og test bildelinken

Internlinker

En spesiell variant av linker er såkalte internlinker på en nettside. Vi kan med disse la brukeren navigere til gitte posisjoner på nettsiden. Wikipedia benytter bl.a. denne teknikken flittig ved å ha en meny for artikkelen øverst i nettsiden. Et klick i menyen nавигerer deg til riktig seksjon lengre ned på nettsiden.

The screenshot shows the Sarpsborg website with a sidebar containing a navigation menu. The menu includes sections like 'Om byen', 'Historie', 'Geograf', 'Tilgang', 'Miljø', 'Utdannelse', 'Kultur', 'Sport', 'Turisme', 'Sosiale medier', 'Bruksanvisninger', 'Kontakt', and 'Om oss'. A red arrow points from the 'Geograf' section in the sidebar to the 'Geograf' section in the main content area, which discusses geographical features of Sarpsborg.

Sarpsborg

Om byen Historie

Kappløp er en av de største løpene i Norge, og har vært arrangert av Sarpsborg idrettsforening siden 1900. Det er en populær løype som går gjennom sentrum av Sarpsborg. Denne løypa er også kjent som "Løpebyens løype".

Geograf

Geograf

Begge deler av landet har gode naturforhold, men det er spesielt i Sør-Norge og i Sørlandet at vi finner de mest interessante naturattraksjonene. Her finner du informasjon om flere av Norges mest kjente naturattraksjoner, fra fjell til sjø.

Natur

Natur

Oppslag om Sarpsborg kommune og dens historie, kultur, miljø, utdannelse, næringsliv, sport og annet.

Selv posisjonen det skal kunne navigeres til er ganske enkelt en tagg med en **id**-attributt. For eksempel en **<div>**-tagg:

```
<div id="kontaktinformasjon">  
</div>
```

Når vi så skal lage en link som navigerer brukeren til denne posisjonen, setter vi href til å være id-en til elementet med et #-tegn foran:

```
<a href="#kontaktinformasjon">Gå til kontaktinformasjon</a>
```

Det er også mulig å lage en link til en viss posisjon på en annen nettside. Dette gjøres i utgangspunktet på samme måte, men vi må ha med filnavnet til nettsiden før #-tegnet.

```
<a href="omoss.html#kontaktinformasjon">Gå til kontaktinformasjon</a>
```

Tabeller

Vi kan strukturere data i tabellform ved hjelp av taggene **<table>**, **<tr>**, **<td>** og **<th>**.

Alle tabeller må ha en **<table>** og **</table>** som omslutter hele tabellstrukturen. Inne i disse taggene har vi så en **<tr>** og **</tr>** for hver rad med data vi ønsker presentere. Tabellceller (felter) markeres med **<td>** og **</td>**.

En tabell med to rader og tre celler i hver rad vil derfor ha følgende struktur:

```
<table>  
  <tr><td>Ole Olsen</td><td>32</td><td>64</td></tr>  
  <tr><td>Per Persen</td><td>56</td><td>75</td></tr>  
</table>
```

Resultatet i nettleseren blir følgende:

Ole Olsen 32 64
Per Persen 56 75

Tabell-strukturen i HTML skal kun benyttes for å presentere data som har en mening i tabellform. Tidligere ble den mye brukt for å lage struktur og design på nettsider. Dette skal imidlertid gjøres ved hjelp av CSS.



Som standard vises tabeller uten kantlinjer. Det er derfor litt vanskelig å få en oppfattelse av at det faktisk er en tabell. Du kan skru på kantlinjer ved å legge inn følgende kode i et internt eller eksternt stilark. Vi kommer til å vise resten av tabelleksemplene med kantlinjer påskrudd i dette kapittelet.

```
table, th, td {
    border: 1px solid black;
}
```

Med kantlinjer er det lettere å se at det faktisk er en tabell:

Ole Olsen	32	64
Per Persen	56	75

Taggen `<td>` er en forkortelse for *table data*, og betyr nettopp data i tabellen. En variant av `<td>` er taggen `<th>` som forteller at innholdet er en *table header*, eller med andre ord er en celle som forteller noe om andre celler i samme rad/kolonne. Vi kan utvide tabellen vår med en ekstra header-rad på topp, samt sette deltakernavnet til å være en header først i hver rad:

```
<table>
    <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
    <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
    <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
</table>
```

Navn	Konkurranse 1	Konkurranse2
Ole Olsen	32	64
Per Persen	56	75

TIPS

Ved bruk av `<th>` legger nettleseren på sitt standard stilark som er fet skrift og midtstilt. Dette kan vi overstyre med CSS selv, om vi ikke ønsker denne visningen.

Forsøk selv:

- Åpne mal-fila og lagre som *tabell.html* i mappa *kapittel3*
- Skriv inn koden under i `<body>`-delen:

```
<table>
    <tr><td>Ole Olsen</td><td>32</td><td>64</td></tr>
    <tr><td>Per Persen</td><td>56</td><td>75</td></tr>
</table>
```

- Lagre og forhåndsvis i nettleser
- Endre linken til stilark slik at den ser ut som følgende:

```
<link rel="stylesheet" type="text/css" href="tabellstil.css" />
```

5. Opprett et nytt dokument i din kodeeditor og skriv inn koden under:

```
table, th, td {
    border: 1px solid black;
}
```

6. Lagre med navnet *tabellstil.css*
 7. Forhåndvis *tabell.html* i nettleser
 8. Endre koden i *tabell.html* som vist:

```
<table>
    <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
    <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
    <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
</table>
```

9. Lagre fila og åpne i nettleser

Tabellceller som dekker flere rader/kolonner

I noen tilfeller får vi behov for en tabellcelle som kan strekkes over flere kolonner eller rader. HTML støtter dette gjennom attributtene **colspan** og **rowspan**.

Vi utvider eksempelet med en deltaker (*Lise Nilsen*) som ble diskvalifisert, og en deltaker (*Trude Hansen*) som fikk to forsøk.

```
<table>
    <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
    <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
    <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
    <tr><th>Lise Nilsen</th><td colspan="2">Diskvalifisert</td></tr>
    <tr><th rowspan="2">Trude Hansen</th><td>23</td><td>44</td></tr>
    <tr><td>93</td><td>16</td></tr>
</table>
```

Merk deg hvordan vi unlater å ta med de **<td>**- og **<th>**-cellene som alt er dekket opp av en **colspan**/**rowspan**. Raden med *Lise Nilsen* har altså bare to celler fordi den andre cellen dekker to plasser. Den andre raden til *Trude Hansen* mangler hele cellen med navnet, da denne «henger igjen» fra raden over.

Navn	Konkurranse 1	Konkurranse2
Ole Olsen	32	64
Per Persen	56	75
Lise Nilsen	Diskvalifisert	
Trude Hansen	23	44
	93	16



Forsøk selv:

1. Fortsett med fila *tabell.html*

2. Legg til følgende kode:

```
<table>
    <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
    <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
    <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
    <tr><th>Lise Nilsen</th><td colspan="2">Diskvalifisert</td></tr>
    <tr><th rowspan="2">Trude Hansen</th><td>23</td><td>44</td></tr>
    <tr><td>93</td><td>16</td></tr>
</table>
```

3. Lagre og forhåndsvis i nettleser

Lister

For å presentere data i listeform har HTML definert de tre taggene ``, `` og ``.

Først må vi velge om vi ønsker en nummerert liste (*ordered list*) ved hjelp av taggen `` eller en punktliste (*unordered list*) ved hjelp av taggen ``. Deretter er det bare å fylle taggen med elementer omsluttet av `` og ``. En uordnet liste kan se slik ut:

<pre> Blå Gul Grå Grønn </pre>	<ul style="list-style-type: none"> • Blå • Gul • Grå • Grønn
---	--

Mens en ordnet liste vil få følgende kode og utseende:

<pre> Blå Gul Grå Grønn </pre>	<ol style="list-style-type: none"> 1. Blå 2. Gul 3. Grå 4. Grønn
---	--

MERK Det er viktig å velge korrekt type liste, da bl.a. søkemotorer benytter informasjonen til å indeksere nettsiden. At to begreper står etter hverandre i en ordered list har betydning, men ikke i en unordered list.

En enkel tommelfingerregel for å avgjøre om du skal benytte `` eller `` er om det har noen betydning at elementene stokkes om. Fremgangsmåter og resultatlister er typisk `` mens lister over medlemmer i en forening og fag du har hatt er ofte ``.



Forsøk selv:

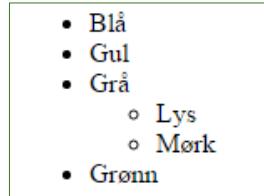
1. Åpne mal-fila og lagre som *liste.html* i mappa *kapittel3*
2. Legg inn denne koden i `<body>`-delen:


```
<ul>
    <li>Blå</li>
    <li>Gul</li>
    <li>Grå</li>
    <li>Grønn</li>
  </ul>
```
3. Lagre og test i nettleser

Nestede lister

Vi kan også neste lister i HTML. Det vil si at enkelte listepunkter inneholder underpunkter. Dette gjøres ved at en del av dataene til ``-elementet er en ny liste.

```
<ul>
  <li>Blå</li>
  <li>Gul</li>
  <li>Grå
    <ul>
      <li>Lys</li>
      <li>Mørk</li>
    </ul>
  </li>
  <li>Grønn</li>
</ul>
```



En vanlig feil er å legge den nye underlista i mellom to ``-elementer:

```
<li>Grå</li>
<ul>
  <li>Lys</li>
  <li>Mørk</li>
</ul>
<li>Grønn</li>
```





Forsøk selv:

- Legg til koden for en nestet liste i fila *liste.html*

```
<ul>
    <li>Blå</li>
    <li>Gul</li>
    <li>Grå
        <ul>
            <li>Lys</li>
            <li>Mørk</li>
        </ul>
    </li>
    <li>Grønn</li>
</ul>
```

- Lagre og forhåndsvis i nettleseren

Definisjonslister

En spesiell variant av lister er såkalte definisjonslister. Dette er lister som består av oppslagsord og forklaring. Dette benyttes oftest i forbindelse med ordlister eller begrepsforklaringer, men det kan også benyttes til å presentere f.eks. navn og tilhørende beskrivelse av personen i en liste.

```
<dl>
    <dt>HTML</dt>
    <dd>Språk for å beskrive innhold og struktur i en nettside</dd>
    <dt>CSS</dt>
    <dd>Språk for å beskrive utseende til en nettside</dd>
    <dt>JavaScript</dt>
    <dd>Språk for å lage interaktivitet i en nettside</dd>
</dl>
```

HTML

Språk for å beskrive innhold og struktur i en nettside

CSS

Språk for å beskrive utseende til en nettside

JavaScript

Språk for å lage interaktivitet i en nettside

Som du ser lages en definisjonsliste ved å omslutte hele listen med en **<dl>**-tagg og deretter legge inn par av taggene **<dt>** (*definition term*) og **<dd>** (*definition data*).

EKSEMPEL - Lage en timeplan

I dette eksemplet skal du lage en nettside som viser en timeplan.



- Opprett en mappe som du kaller *timeplan* i mappa *kapittel2*. Alle filene du lager i dette eksemplet skal lagres i denne mappen
- Åpne mal-fila og endre koden slik at den blir som vist under. Filen skal lagres med navnet *index.htm*

```
<!DOCTYPE html>
<html>
    <head>
        <title>Timeplan</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="stil.css" />
    </head>

    <body>

        <h1>Timeplan</h1>

        <table>
            <tr class="dag">
                <th></th>
                <th>Mandag</th>
                <th>Tirsdag</th>
                <th>Onsdag</th>
                <th>Torsdag</th>
                <th>Fredag</th>
            </tr>
            <tr>
                <td class="tid">0800 - 0845</td>
                <td>Kroppsøving</td>
                <td>Norsk</td>
                <td>Engelsk</td>
                <td>IT-1</td>
                <td>Kjemi</td>
            </tr>
            <tr>
                <td class="tid">0900 - 0945</td>
                <td>Matte</td>
                <td>Norsk</td>
                <td>Engelsk</td>
                <td>Historie</td>
                <td>Kjemi</td>
            </tr>
        </table>

    </body>
</html>
```



3. Lag en ny fil som du kaller *stil.css* og skriv inn følgende kode:

```
table, th, td {
    border: 1px solid black;
}
```

4. Sjekk resultatet i nettleseren:

Vi har foreløpig bare lagd to rader med fag. Etter hvert skal du selv legge inn flere rader slik at vi får en fullstendig timeplan.

Timeplan

	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
0800 - 0845	Kroppsøving	Norsk	Engelsk	IT-1	Kjemi
0900 - 0945	Matte	Norsk	Engelsk	Historie	Kjemi

5. Vi ser at noen fag går over to (eller flere) timer. I stedet for å repetere faget i flere celler kan vi slå dem sammen ved hjelp av **rowspan**. Endre innholdet i **<table>**-taggen slik at vi får:

```
<table>
    <tr class="dag">
        <th></th>
        <th>Mandag</th>
        <th>Tirsdag</th>
        <th>Onsdag</th>
        <th>Torsdag</th>
        <th>Fredag</th>
    </tr>
    <tr>
        <td class="tid">0800 - 0845</td>
        <td>Kroppsøving</td>
        <td rowspan="2">Norsk</td>
        <td rowspan="2">Engelsk</td>
        <td>IT-1</td>
        <td rowspan="2">Kjemi</td>
    </tr>
    <tr>
        <td class="tid">0900 - 0945</td>
        <td>Matte</td>
        <td>Historie</td>
    </tr>
</table>
```

Vi har fjernet cellene med norsk, engelsk og kjemi i den siste raden slik at fagene ikke kommer «dobbelt opp».

Timeplan

	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
0800 - 0845	Kroppsøving	Norsk	Engelsk	IT-1	Kjemi
0900 - 0945	Matte			Historie	

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

6. Vi skal endre CSS-fila slik at tabellen får et bedre utseende. Åpne *stil.css* og endre den til:

```
body {
    font-family: sans-serif;
    margin: 20px;
}

table {
    border-collapse: collapse;
    background-color: linen;
    box-shadow: 0px 0px 20px grey;
}

table, th, td {
    border: 1px solid grey;
    padding: 5px;
}

td, th {
    width: 100px
}

.dag {
    background-color: coral;
    border-bottom: 3px solid black;
}

.tid {
    background-color: lemonchiffon;
    font-weight: bold;
    border-right: 3px solid black;
}
```

Vi bruker **border-collapse** for å fjerne de «doble» strekene i tabellen. Vi har også lagd egne CSS-klasser for dag-raden og tids-kolonnen slik at det blir enklere å gjøre utseende på disse forskjellig fra resten av tabellen.



7. Lagre fila med navnet *stil.css* i mappa og sjekk resultatet i nettleseren:

Timeplan

	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
0800 - 0845	Kroppseøving			IT-1	
0900 - 0945	Matte	Norsk	Engelsk	Histone	Kjemi

8. Fortsett med å lage flere rader med fag. Ta gjerne en kopi av den første raden med fag og bruk samme metode som vist ovenfor for å slå sammen celler

Medieinnhold

Inntil HTML5 kom på banen var det et mareritt å sette inn medieinnhold av typen lyd og video i en nettside. Med HTML5 kom imidlertid de to nye taggene `<audio>` og `<video>` som gjør jobben vesentlig enklere.

I utgangspunktet var tanken at både `<audio>`- og `<video>`-taggene skulle fungere mer eller mindre likt som ``-taggen. Dessverre ble det, som vi snart skal se, en uenighet om hvilke formater som skulle støttes, så litt mer jobb er det, selv om likevel ganske enkelt.



Det er viktig at lyd- og videofilene følger HTML-fila til nettsiden og blir referert riktig. Dataene blir altså ikke en del av selve HTML-fila, men refereres til på samme måte som bilder i en ``-tagg også må følge med HTML-fila.

Både lyd og video lar seg spille av med enkle avspillingskontrollere. Ønsker vi å gjøre mer avanserte ting må vi imidlertid programmere i nettsiden ved hjelp av JavaScript. Dette er utenfor denne bokas rammer.

Lyd

For å sette inn en avspiller som brukeren kan se og interaktere med, legger vi inn en `<audio>`-tagg på følgende måte:

```
<audio controls="controls">
  <source src="lydfil.mp3" type="audio/mpeg">
    Nettleseren støtter ikke lydavspilling
</audio>
```



Legg spesielt merke til at vi legger en feilmelding inne i `<audio>`-taggen. Støtter ikke nettleseren lydavspilling vil dette innholdet vises.

En kompliserende del av lyd og nettsider er som tidligere nevnt at ulike nettlesere støtter ulike formater. Når denne boka ble skrevet så lydformat-støtten slik ut for de siste versjonene av nettleserne, men dette forbedres stadig:

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

	Ogg	Wav	MP3
Internet Explorer	X	X	✓
Safari	X	✓	✓
Chrome	✓	✓	✓
Firefox	✓	✓	✓
Opera	✓	✓	✓

Ut i fra denne oversikten kan det være lurt å foreløpig benytte MP3 som lydformat.

Legg spesielt merke til at **<audio>**-taggen kan bestå av flere ulike **source**-tagger. Dermed kan vi legge inn flere ulike formater slik at nettleseren kan velge å benytte det formatet den foretrekker.

```
<audio controls="controls">
  <source src="lydfil.mp3" type="audio/mpeg">
  <source src="lydfil.ogg" type="audio/ogg">
    Nettleseren støtter ikke lydavspilling
</audio>
```

Husk at om du benytter flere ulike lydformater så må du oppdatere alle filene om du senere endrer litt på selve lyden. Det er fort gjort å glemme å gjøre endringen i en av filene, og ikke oppdage feilen da din nettleser foretrekker et av de andre formatene.



Følgende attributter er vanlige å benytte sammen med **<audio>**-taggen:

controls="controls" - angir at vi ønsker ha avspillingskontrollere.

autoplay="autoplay" - angir at vi ønsker at lydavspillingen skal starte så snart nettsiden er lastet.

loop="loop" - angir at vi ønsker at lydavspillingen skal starte på nytt når den kommer til slutten.

TIPS

Du synes kanskje det ser rart ut å ha et attributt som har samme verdi som attributtnavnet. Vi kunne droppet hele verdien, men det er vanlig at alle attributter skal ha en verdi. Siden det ikke er definert noen gyldige verdier, er det vanlig å sette attributt-navnet også som verdi.

Video

På eksakt samme måte som vi setter inn `<audio>`-tagger støtter HTML5 også en `<video>`-tagg. Den eneste store forskjellen er at vi bør angi en størrelse på avspillingen.

```
<video width="640" height="480" controls="controls">
  <source src="videofil.mp4" type="video/mp4">
  <source src="videofil.ogg" type="video/ogg">
  Nettleseren støtter ikke videoavspilling
</video>
```



Vi har det samme problemet med formater når vi benytter video. Ikke alle nettlesere støtter alle formatene. På tidspunktet da denne boka ble utgitt så situasjonen slik ut:

	Ogg	WebM	MP4
Internet Explorer	✗	✗	✓
Safari	✗	✗	✓
Chrome	✓	✓	✓
Firefox	✓	✓	✓
Opera	✓	✓	✓

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Som du ser er MP4 foreløpig det formatet med best støtte, men vi kan som med **<audio>**-taggen ha med flere alternative formater.

Følgende attributter til **<video>**-taggen kan være nyttige:

controls="controls"	- angir om vi ønsker ha avspillingskontrollere
autoplay="autoplay"	- angir at vi ønsker at videoavspillingen skal starte så snart nettsiden er lastet
loop="loop"	- angir at vi ønsker at videoavspillingen skal starte på nytt når den kommer til slutte
poster="url"	- spesifiserer et bilde som skal vises før videoen er lastet/starter. Vil også bli benyttet ved utskrift

Forsök selv:

1. Lag en ny fil i mappa *kapittel3* kalt *video.html* ut i fra mal-fila
2. Legg inn fila *videotest.ogg* fra øvingsfilene til boka
3. Legg inn følgende kode i **<body>**:

```
<video width="640" height="480" controls="controls">
    <source src="videotest.ogg" type="video/ogg">
    Nettleseren støtter ikke videoavspilling
</video>
```

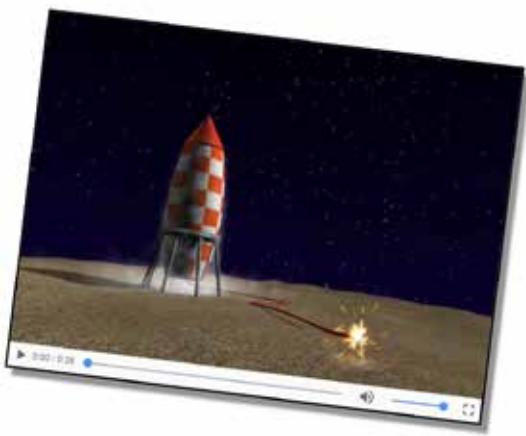
4. Test nettsiden i nettleseren og spill av videoen



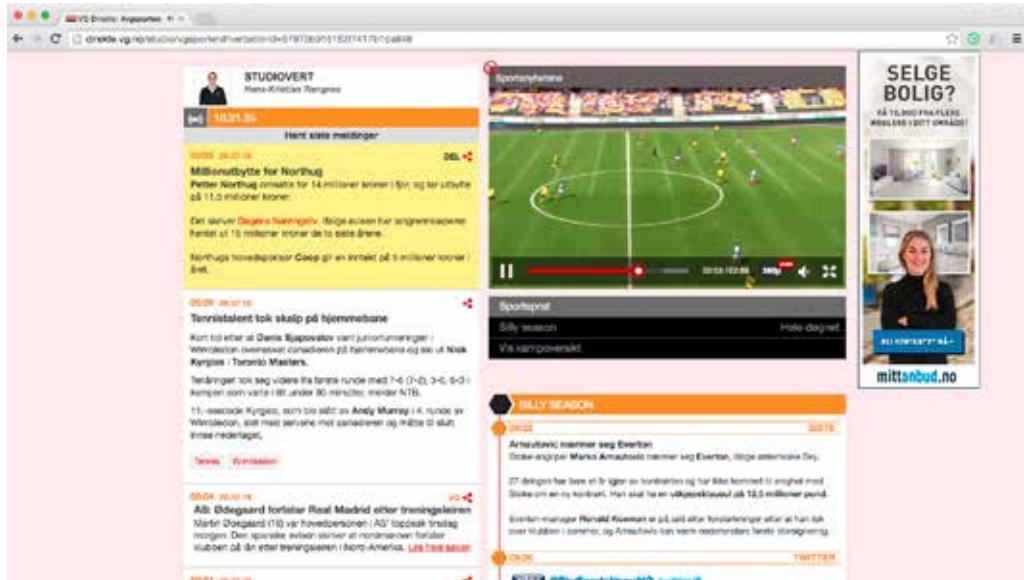
Video og CSS

En litt spennende ting med <video>-taggen er at den er ansett som en boks på samme måte som et hvilket som helst annet element. Dermed kan vi benytte CSS for å stilsette videoavspillingen.

```
video {
    box-shadow: 10px 10px 5px #888888;
    border-style:solid;
    transform: rotate(7deg);
    margin: 15px;
}
```



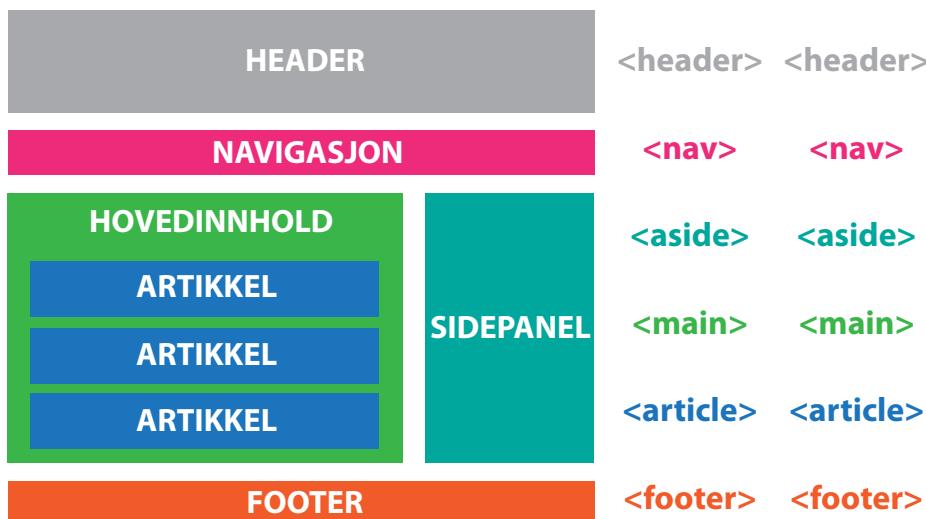
Vi kommer tilbake til detaljene rundt de ulike CSS-egenskapene vi her viste i neste kapittel.



Semantiske tagger

En for mange ukjent side av HTML5 er alle de semantiske taggene som finnes, og som foreløpig ikke har noen direkte «effekt» i visningen av nettsiden. De har derimot en stor effekt på betydningen av innholdet i forhold til maskinell lesing (mer om dette snart), og vi bør derfor være flinke til å benytte dem.

I HTML5 er det et definert mange nye strukturelle tagger, hvorav de vanligste i bruk er `<main>`, `<nav>`, `<header>`, `<footer>`, `<section>`, `<article>` og `<aside>`. Alle disse benyttes for å markere deler av nettsiden som de fleste nettsider består av:



`<main>`

Taggen `<main>` skal benyttes for å markere hva som er hovedinnholdet på nettsiden. I denne taggen skal selve informasjonen vi vil formidle plasseres.

`<section>` og `<article>`

Ofte deles `<main>` igjen inn i `<section>` eller `<article>`-tagger. Det er ikke helt lett å se hva som er forskjellen på `<section>` og `<article>`, men generelt kan vi si at én `<article>` skal handle om ett tema. Et typisk eksempel er en forsida på en blogg, der hvert innlegg er en `<article>`. Taggen `<section>` benyttes først og fremst til å dele inn `<article>` i mindre deler, slik som ingress, hoveddel og konklusjon. En kompliserende faktor er at også `<section>` kan brukes for å samle flere `<article>`-tagger. I disse tilfellene tenker man på `<section>` som seksjoner slik som sport, nyheter og vær i en avis.

<header> og <footer>

Det som tilhører headeren og footeren på nettsiden plasseres i de tilhørende taggene **<header>** og **<footer>**. I headeren ligger typisk logo, mens i footeren kan kontaktinformasjon, copyright og informasjon om når nettsiden sist ble oppdatert ligge. Det er også mulig å benytte taggene **<header>** og **<footer>** inne i f.eks. en **<section>** eller **<article>**.



Pass på å ikke blande sammen taggene **<head>** og **<header>**.

<nav>

Taggen **<nav>** benyttes til å markere den interne navigasjonsmenyen på nettsiden.

<aside>

Innhold som er «på siden» av selve hovedinnholdet til nettsiden plasseres i taggen **<aside>**. I følge den opprinnelige utgaven av HTML-spesifikasjonen skal imidlertid innholdet være relatert til teksten. En **<aside>** kan derfor typisk benyttes til «faktabokser» som f.eks. utdyping begreper benyttet i teksten.

Senere har koblingen mot hovedinnholdet blitt tonet en del ned i spesifikasjonen, så det er også greit å benytte **<aside>** til f.eks. annonser, liste over andre blogginnlegg og søkefelt.

Hvorfor benytte tagger som ikke har noen visuell effekt?

Hvorfor skal vi så bruke tagger som tilsynelatende ikke gjør noe? Det enkle svaret er at mennesker er flinke på å tolke innhold og forstå betydningen av tekst - det er imidlertid ikke maskiner. Ønsker vi at maskiner (typisk nettleseren) skal kunne gjøre valg og gi ulike muligheter ut i fra hvordan nettsiden er bygget opp, så må vi fortelle den hva som er hva.

Søkemotorer, og annen maskinell tolkning slik som talesyntese, benytter flittig disse taggene for å forstå oppbyggingen av en nettside. Ved hjelp av taggene kan de avgjøre hva som er header, footer, meny, hovedinnhold osv.

Med tiden vil det også komme flere funksjoner i nettleserene som benytter disse taggene. Tenk deg hvor praktisk det ville være om en utskrift av nettsiden kun skriver ut innholdet i **<main>**-taggen, eventuelt med **<header>** og **<footer>** på hvert ark. Hvor praktisk ville det ikke også være at nettleseren på mobilen skjulte menyen inntil vi trykket på en meny-knapp eller gjorde en såkalt gesture på selve telefonen?

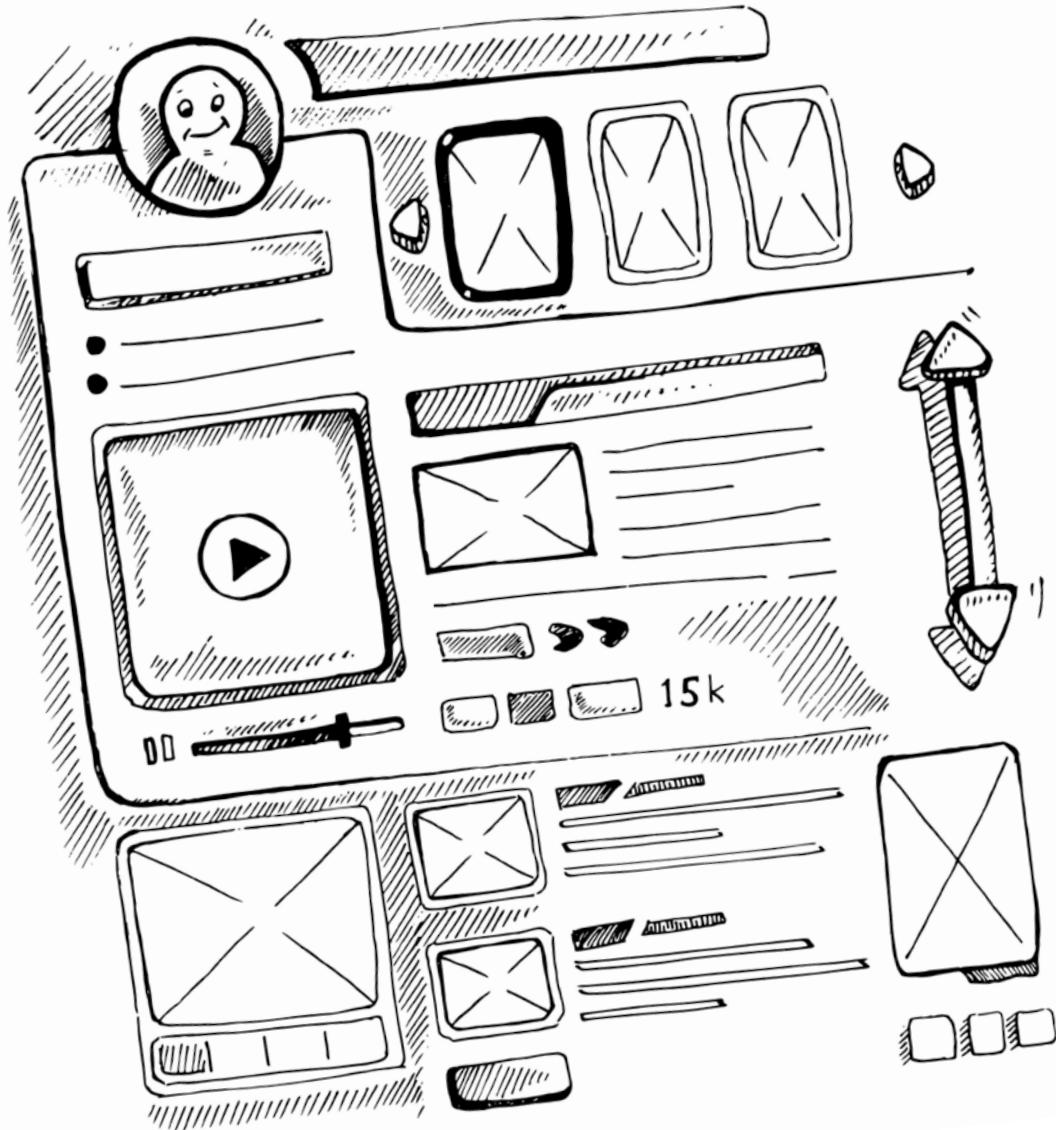
KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Sist, men ikke minst, så vil disse taggene ofte representere bokser som vi ønsker stilsette. Hadde vi ikke benyttet de semantiske taggene ville vi som regel endt opp med å uansett lage `<div>`-elementer med en `id` satt til noe lignende.

```
<div id="meny">  
  ...  
</div>
```

Det var faktisk slik de som utvikler HTML-standarden fant ut hvilke semantiske tagger man skulle lage. De så på de vanligste id-navnene på `<div>`-tagger i nettsider: <https://developers.google.com/webmasters/state-of-the-web/2005/classes>

TIPS



4 Formgi nettsider

I dette kapitlet vil du lære

- om ulike typer selectorer
- om måleenheter
- om fargeangivelser
- å stilsette tekst
- å sette inn bilder ved hjelp av CSS
- å stilsette CSS-bokser
- om nettleserprefiks

Til nå har vi sett på hovedprinsippene for hvordan CSS fungerer og er bygget opp. Vi har også sett noen eksempler på egenskaper og verdier. I dette kapittelet skal vi gå mye mer grundig inn på hvilke muligheter CSS gir oss. Som vi skal se finnes det en rekke ulike måter å angi selectorer på, noe som gir oss mulighet til å endre på helt bestemte elementer i nettsiden. Vi skal også ta for oss ulike måter å justere tekst og farger på, samt hvilke egenskaper som er knyttet til box model.



Selectorer

Vi har tidligere sett på tre typer selectorer som kan benyttes i CSS for å velge hvilke elementer som CSS-reglene vi skriver skal gjelde er. Disse tre typene er:

1. Bruk av taggnavn:

```
p {  
    color:blue;  
}
```

2. Henvisning til en id:

```
#ingress {  
    font-weight: bold;  
}
```

3. Bruk av en klasse:

```
.brodtekst {  
    font-style: italic;  
}
```

I tillegg til disse tre typene har CSS også en rekke andre selectorer, som hjelper oss til å gjøre helt spesifikke utvalg elementer som vi så kan definere utseende på.

Eksempelvis kan vi velge at stilreglene kun skal gjelde de gangene en klasse er brukt på en spesiell tagg. Her ser du at regelen kun skal gjelde de gangene som klassen viktig er benyttet på en link:

```
a.viktig {  
    font-weight: bold;  
}
```

Vi skal i denne seksjonen ta for oss de vanligste måtene å bygge opp selectorer på. Å beherske de ulike selectorene vil medføre at vi kan lage mer elegant og enklere CSS-kode. Riktig bruk av selectorer vil også gjøre at CSS-koden fungerer selv når HTML-dokumentet endres.

Sammensatte selectorer

Dersom vi ønsker at samme regel skal gjelde flere selectorer, kan vi liste de opp med komma mellom hver selector. Følgende regel om rød farge vil i dette tilfellet gjelde paragrafer, linker og listeelementer:

```
p, a, li {  
    color: red;  
}
```

Alle selectortyper er tillatt, og vi kan blande ulike typer (slik som *tagg*, *id* og *klasse*):

```
a, #ingress, .brodtekst {
    color: red;
}
```

I stedet for å liste opp alle tagger i HTML, kan vi benytte det spesielle tegnet * om vi ønsker noe skal gjelde alle elementer:

```
* {
    color: red;
}
```

Relasjoner og selectorer

Vi kan begrense utvalget ytterligere ved å angi at et gitt element kun skal endres dersom det står i en bestemt relasjon til et annet element på nettsiden. En -tagg som står «inne i» en <a>-tagg er et eksempel på en slik relasjon, og medfører som du kanskje husker at bildet da vil fungere som en link.

```
<a href="http://www.gyldendal.no">
    
</a>
```

Den enkleste måten å angi en slik relasjoner er å liste opp to elementer med et mellomrom mellom:

```
a img {
    border-style: solid;
    border-width: 1px;
    border-color: blue;
}
```

 Det er fort gjort å tenke feil rekkefølge når vi benytter skrivemåten med mellomrom mellom to selectorer. Det er den siste som skal stå «inni» den første for at regelen skal gjelde. Vi kan huske på at vi skal lese det «bakvendt», slik som «img i a».

En annen metode for å angi relasjon mellom selectorer er å bruke + tegnet. Et eksempel på dette er :

```
h2 + p {
    font-style: italic;
}
```

I dette eksempelet vil alle paragrafer som kommer direkte etter en <h2>-tagg bli satt i kursiv. En paragraf som kommer etter en annen type tagg vil ikke bli satt i kursiv.

Det finnes en variant av denne selectoren, som ikke krever at taggen kommer direkte etter, men bare etter. Altså at det har forekommert en tagg av en viss type først. F.eks. vil følgende selector gjelde alle paragrafer etter at det først har forekommert en `<h2>`-tagg i HTML-dokumentet: `h2 ~ p`



Forsøk selv:

1. Opprett en mappe med navn *kapittel4*
2. Åpne mal-filen i din kodeeditor og skriv inn følgende tekst i `<body>`-delen:

```
<h2>Skolen min</h2>
<p>Min skole ligger midt i byen og har 300 elever.</p>
<p>Skolen tilbyr både studiespesialiserende og yrkesfaglige.</p>
<p>Det er flere tilbud til elever etter skoletid, slik som leksehjelp og
styrketrening.</p>
```
3. Endre linken til stilark slik at den referer til *selectorer1stil.css*:

```
<link rel="stylesheet" type="text/css" href="stilark1.css" />
```
4. Lagre med navnet *selectorer1.html* i mappen *kapittel4*
5. Opprett en ny fil i kodeeditoren og skriv inn følgende CSS-kode:

```
p {
    font-style: italic;
}
```
6. Lagre med navnet *selectorer1stil.css*
7. Forhåndsvise *selectorer1.html* i nettleseren
8. Endre *selectorer1stil.css* slik at koden ser slik ut:

```
h2 + p {
    font-style: italic;
}
```
9. Lagre og sjekk nå hvordan *selectorer1.html* ser ut i nettleseren
10. Endre koden igjen slik at den ser slik ut:

```
h2 ~ p {
    font-style: italic;
}
```
11. Lagre og forhåndsvise i nettleser

Selectorer på attributter

I noen tilfeller kan det også være nyttig å velge elementer basert på om de har et bestemt attributt eller om dette attributtet har en bestemt verdi. Følgende vil f.eks. sette en blå ramme rundt alle bilder som har en `alt`-attributt:

```
img[alt] {  
    border-style: solid;  
    border-width: 1px;  
    border-color: blue;  
}
```

Vi kan også gjøre en sjekk på selve verdien. Følgende kode vil lage en strek over, i stedet for under, linker som er absolutte (altså starter med `http`):

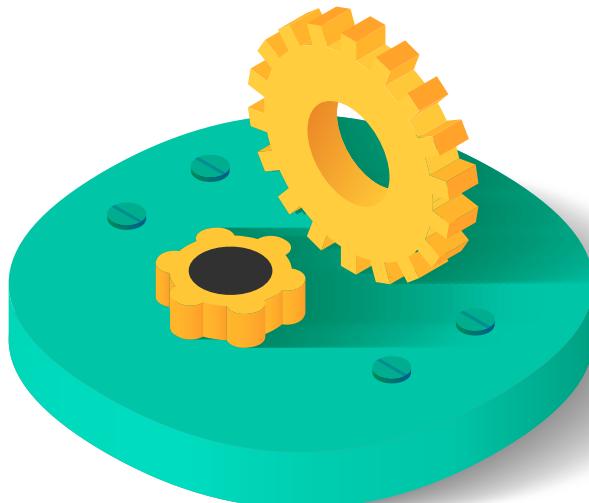
```
a[href^="http"] {  
    text-decoration: overline;  
}
```

Eller vi kan fargelegge alle linker som peker til dokumenter av typen PDF med rød skrift ved å sjekke at verdien i `href` ender på `.pdf`:

```
a[href$=".pdf"] {  
    color:red;  
}
```

De mest brukte slike attributt-selectorene er:

<code>[attributtnavn]</code>	Har attributtet
<code>[attributtnavn^=verdi]</code>	Starter med verdien
<code>[attributtnavn\$=verdi]</code>	Slutter med verdien
<code>[attributtnavn=verdi]</code>	Er lik verdien
<code>[attributtnavn*=verdi]</code>	Inneholder verdien



Forsøk selv:

1. Åpne mal-filen i din kodeeditor og skriv inn følgende tekst i <body>-delen:

```
<h1>Bildegalleri</h1>


```



2. La HTML-koden referere til et stilark med filnavn *selectorer2stil.css*
3. Lagre med navnet *selectorer2.html*
4. Sørg for at øvingsfilene *frosk.jpg* og *frosk3.jpg* ligger i mappen *kapittel4*
5. Vis nettsiden i nettleseren
6. Opprett en ny fil i din kodeeditor, gi den navnet *selectorer2stil.css* og skriv inn følgende kode:

```
img[alt] {
    border-style: solid;
    border-width: 1px;
    border-color: blue;
}
```

7. Lagre og se nå på *selectorer2.html* i nettleseren din

Som du ser får det siste bildet en blå ramme rundt – dette fordi dette har attributtet alt i -taggen.

8. Endre koden i *selectorer2.html* slik at også bildet *frosk.jpg* blir vist med en blå ramme rundt

Pseudoelementer

I CSS er det også definert en del selectortyper som velger deler av et element. Disse har fått navnet *pseudoelementer*. Vi kan f.eks. velge å gjøre noe med første linje i en paragraf ved å benytte pseudoelementer **:first-line**:

```
p::first-line {
    font-style: italic;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin aliquam dolor eget nisl vestibulum, eu lobortis lacus fringilla. Duis congue fringilla urna.

Eller kanskje gjøre første bokstav litt ekstra kunstnerisk utformet, slik de er i gamle bøker, gjennom pseudoelementet `::first-letter`:

```
p::first-letter {
    font-size: 200%;
    font-weight: bold;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin aliquam dolor eget nisl vestibulum, eu lobortis lacus fringilla. Duis congue fringilla urna.

To veldig spesielle pseudoelementer er `::before` og `::after`. Her kan vi i CSS sammen med egenskapen `content` sette inn tekst før og etter et element, dersom denne teksten er for design å regne. F.eks. kan vi sette inn tegnene -- før og etter alle `<h2>`-overskrifter:

```
h2::before {
    content: "--";
}

h2::after {
    content: "--";
}
```

--Dette er første overskrift--

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

--Dette er andre overskrift--

Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Legg merke til at de ekstra tegnene vi setter inn ved hjelp av `content` i CSS ikke blir en del av innholdet, men av designet. Søkemotorer vil f.eks. ikke bry seg om denne informasjonen.

Tabellen under lister de vanligste pseudoelementene.

<code>::first-letter</code>	Første bokstav
<code>::first-line</code>	Første linje
<code>::selection</code>	Markert tekst
<code>::before</code>	Tekst før elementet
<code>::after</code>	Tekst etter elementet

Pseudoklasser

Pseudoklasser benyttes for å begrense et utvalg av elementer. Tenk deg at du har en liste med flere elementer (-taggen), og ønsker at kun det første elementet skal ha fet skrift. Ved bruk av pseudoklasse kan du angi dette på følgende måte i CSS-arket:

```
li:first-child {
    font-weight: bold;
}
```

1. Blå
2. Gul
3. Grå
4. Grønn

Underelementer (altså et element inne i et annet) omtales som et «barn». Det høres imidlertid mye bedre ut med det engelske begrepet *child element*. Tilsvarende omtales elementet som omslutter underelementene som *foreldrelement* eller *parent element*.

TIPS

Ønsker du at det bare skal gjelde det tredje element i listen blir koden slik:

```
li:nth-child(3) {
    font-weight: bold;
}
```

1. Blå
2. Gul
3. Grå
4. Grønn

Pseudoklassen :nth-child kan også jobbe med andre ting enn nummerering, og begrepene **even** og **odd** er kjekke å ha for å gjøre noe med annen hvert element. Vi kan f.eks. fargelegge radene i en tabell annenhver gang grå og hvite:

```
tr:nth-child(odd) {
    background-color: #aaa;
}

tr:nth-child(even) {
    background-color: #fff;
```

Ole Olsen	32	64
Per Persen	56	75
Lise Nilsen	23	44
Truls Trulsen	93	16

Husk at pseudoelementer (se eget avsnitt) starter med to kolon, mens pseudoklasser starter med ett kolon.

MERK

De vanligste pseudoklassene er listet i tabellen under. De kan brukes både med og uten en selector foran kolonet.

:first-of-type	Første gang elementet dukker opp i nettsiden
:last-of-type	Siste gang elementet dukker opp i nettsiden
:only-of-type	Dersom elementet kun finnes en gang i nettsiden
:only-child	Dersom elementet er det eneste «barnet» inne i et annet element
:nth-child(x)	Angir barn nummer X under et element
:nth-last-child(x)	Angir barn nummer X (telt fra slutten) under et element
:nth-of-type(x)	Angir element nummer X av denne typen i hele nettsiden
:nth-last-of-type(x)	Angir element nummer X (telt bakfra) av denne typen i hele nettsiden
:first-child	Angir første barn under et element
:last-child	Angir siste barn under et element
:empty	Angir element uten innhold

Det finnes også en annen gruppe av pseudoklasser som forteller noe om status til et element. Status kan f.eks. være om elementet er valgt eller har fokus. Vi kan f.eks gjøre noe med utseende til linker som er besøkt av brukeren tidligere på følgende måte:

```
a:visited {
    background-color: red;
}
```

De vanligste pseudoklassene av denne typen finner du i tabellen under:

a:visited	Link som er besøkt
a:link	Link som ennå ikke er besøkt
a:active	Linken i det vi klikker på den (og holder museknappen nede)
:hover	Elementet i det vi holder musepekeren over den
:target	Elementet vi linker til (internlink på den samme nettsida)



Forsøk selv:

- Åpne mal-filen i din kodeeditor og skriv inn følgende kode i **<body>**-delen

```
<table>
    <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
    <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
    <tr><th>Lise Nilsen</th><td>23</td><td>44</td></tr>
    <tr><th>Truls Trulsen</th><td>93</td><td>16</td></tr>
</table>
```



- Endre stilarkreferansen til *selectorer3stil.css*
- Lagre fila med navnet *selectorer3.html* i mappa *kapittel4*
- Opprett en ny fil i kodeeditoren og skriv inn følgende CSS-kode:

```
table, th, td {
    border: 1px solid black;
}
```

- Lagre som *selectorer3stil.css* og se på *selectorer3.html* i nettleseren din
- Legg til følgende kode i CSS-fila og se på *selectorer3.html* i nettleseren

```
tr:nth-child(odd) {
    background-color: #aaa;
}

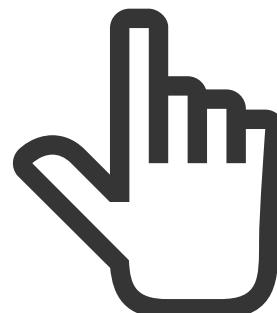
tr:nth-child(even) {
    background-color: #fff;
}
```

EKSEMPEL - interaktivitet



I dette eksemplet skal du lage en nettside som viser ulike artikler når du klikker på linker som er utformet som knapper.

- Opprett en mappe som du gir navnet *interaktivitet*. Alle filene du lager i dette eksemplet skal lagres i denne mappa



2. Start din kodeeditor og skriv inn følgende tekst:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Interaktivitet</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stil.css" />
  </head>

  <body>
    <nav>
      <a href="#Appelsin" class="knapp">Appelsin</a>
      <a href="#Eple" class="knapp">Eple</a>
      <a href="#Mango" class="knapp">Mango</a>
    </nav>

    <article id="Appelsin">
      <h2>Appelsin</h2>
      <p>...masse tekst her...</p>
    </article>
    <article id="Eple">
      <h2>Eple</h2>
      <p>...masse tekst her...</p>
    </article>
    <article id="Mango">
      <h2>Mango</h2>
      <p>...masse tekst her...</p>
    </article>
  </body>
</html>
```

Der det står *...masse tekst her...* kan du bytte ut med egen tekst. Legg merke til at **<a>**-taggene (knappene) har internlinker til ulike artikler på siden. I tillegg har vi angitt en CSS-klasse kalt *knapp* slik at vi lettere kan bestemme utseende på dem i CSS-koden.

3. Lagre dokumentet i mappa med filnavnet *index.html* og sjekk ut sida i nettleseren
 4. Vi skal nå lage CSS-fila. Velg **Fil > Ny** i din kodeeditor og lagre fila som *stil.css*. Skriv inn følgende kode og lagre på nytt:

```
body {
  background-color: lightgrey;
  font-family: sans-serif;
}

.knapp {
  display: inline-block;
  width: 100px;
  text-align: center;
  text-decoration: none;
  color: black;
  padding: 5px;
```



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
background-color: darkgrey;
font-weight: bold;
border: solid 1px black;
cursor: pointer;
}
```

Legg merke til at vi bruker **display: inline-block** for å kunne angi en fast bredde på knappene – i tillegg til at de skal ligge ved siden av hverandre. Du kan lese mer om **display** senere i dette kapittelet. Vi bruker **cursor: pointer** slik at vi ser en hånd når musa er over en knapp. For å fjerne understreken som vanligvis vises i en link bruker vi **text-decoration: none**.

5. Vi ønsker at bakgrunnsfargen på knappene skal endre seg når vi beveger musa over dem. Legg til følgende kode i CSS-fila:

```
.knapp:hover {
    background-color: lightyellow;
}
```

Sjekk at at knappene forandrer farge når du flytter musa over dem.

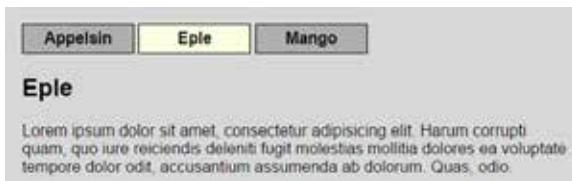
6. Når vi klikker på en knapp skal tilsvarende artikkel vises nedenfor. Vi begynner med å skjule alle artiklene ved å bruke **display:none**. Legg til følgende kode i CSS-fila:

```
article {
    display: none;
}
```

7. Når vi klikker på en knapp skal tilsvarende artikkel vises nedenfor. Vi gjør dette ved å bruke pseudoklassen :**target**. Pseudoklassen :**target** gjør at vi kan forandre stil på elementer som vi linker til. I vårt tilfelle er det artiklene vi linker til når vi klikker på knappene. Vi legger derfor til følgende kode i CSS-fila slik artiklene blir synlige igjen (får tilbake vanlig – *initial* – visning):

```
article:target {
    display: initial;
}
```

Sjekk at riktig artikkel vises når du klikker på knappene:



Måleenheter

CSS tilbyr mange ulike typer måleenheter som kan brukes til angivelse av størrelse og plassering av elementer. Dersom vi vil at alle avsnitt skal ha en marg på 30 piksler skriver vi følgende i CSS-fila:

```
p {
    margin: 30px;
}
```

De ulike måleenhetene grupperes vanligvis inn i relative- og absolutte måleneheteter.

De absolutte måleenhetene tilsvarer faktiske mål:

- cm** Centimeter på visningsmediumet
- mm** Millimeter på visningsmediumet
- in** Tommer på visningsmediumet
- pt** Punkter på visningsmediumet. Det er 72 punkter i en tomme
- pc** Picas på visningsmediumet. Det er 12 punkter i en pica
- px** Måleenheten px er en litt rar, men veldig mye benyttet enhet. Den har ingen gitt størrelse, men avhenger av mediet. For medier med lav oppløsning er **1px** det samme som 1 piksel, men for medier med høy oppløsning er **1px** ofte flere piksler



Det er stort sett bare ved utskrift at de absolutte måleenhetene fungerer som eksakte mål. Du kan f.eks. ikke regne med at et visst antall **cm** vil vises korrekt på en skjerm. Derfor er ikke disse måleenhetene anbefalt for bruk på skjerm, med unntak av **px**.

De relative måleenhetene angir størrelser ut i fra andre elementer:

- em** Angis i forhold til fontstørrelsen. Verdien **1.5em** tilsvarer f.eks. en og en halv gang fontstørrelsen
- %** Angis i forhold til foreldre-elementet. **100%** er samme størrelse som foreldre-elementet
- vw** Angis i forhold til *viewport* (området av nettsiden vi ser) sin bredde. Målet **100vw** tilsvarer viewport sin bredde. Målet **150vw** tilsvarer en og en halv gang viewport sin bredde. I en nettleser er viewport den synlige flaten. Ved utskrift er viewport den delen av arket printeren kan skrive på
- vh** Tilsvarende som **vw**, men angis i forhold til viewport sin høyde

Dersom du ønsker å angi verdien 0 er det irrelevant hvilken måleenhet vi angir det i, da 0 betyr det samme i alle måleenheter. Det er derfor vanlig å droppe måleenhet når man angir 0 som verdi: **margin-left: 0;**



Måleenheten skrives helt inntil tallverdien, slik som **45%, 20vh** og **70px**.

Ettersom vi i dag stort sett alltid ønsker å lage nettsider med skalerbart design er det viktig å forsøke unngå absolute måleenheter og heller basere nettsiden på relative måleenheter. Det er ofte vanskeligere å lage gode design med relative måleenheter, men når vi først får det til å fungere med skalering vil nettsiden tilpasses de fleste skjermstørrelser.

Farger

Til nå har vi basert oss på å angi farger ut i fra de 140 ferdigdefinerte fargenavnene i CSS, slik som *red*, *green*, *navy*, *gold*, *honeydew* osv.



Du kan finne en oversikt over alle fargenavnene, samt en forhåndsvisning, i CSSspesifikasjonen: <https://www.w3.org/TR/css3-color/#svg-color>

I noen tilfeller er imidlertid ikke disse 140 fargenavnene nok, og vi ønsker å «blande» vår egen farge.

Den vanligste metoden er å angi hvor mye rød, grønt og blått vi vil at vår farge skal bestå av. Angir vi maksimalt av alle tre farger får vi hvitt, og angir vi minimalt av alle tre farger får vi sort. Dette er da etter den såkalte RGB-fargeangivelsen. I CSS kan vi angi dette på mange måter. Vi kan f.eks. angi dette som prosentandeler:

```
p {  
    color: rgb(75%, 43%, 100%); /* Lilla farge på teksten */  
}
```

Vanligere er det nok å angi verdiene fra 0 til 255 (255 er et «rundt tall» i det binære tallsystemet):

```
p {  
    color: rgb(191, 110, 255); /* Lilla farge på teksten */  
}
```



Fra tidlig i nettsiders historie var det også vanlig å angi tallverdiene fra 0 til 255 som *heksadesimale* tallverdier, og dette vil du fortsatt møte i CSS-kode. Dette er et tallsystem der man i stedet for å telle fra 0 til 10 teller fra 0 til f, så altså 1,2, … , 8,9,a,b,c,d,e,f. Verdien c tilsvarer dermed 12 i titallssystemet. Verdien ff er det samme som 255: . Vi setter så sammen tre grupper med to siffer hver, for å angi rødt, grønt og blått:

```
p {
    color: #bf6eff;      /* Lillafarge på teksten */
}
```

Dersom man kun benytter «like tall» i det heksadesimale systemet, slik som 55, aa og ff, kan de også skrives som et enkelt siffer. Verdien #55aaff kan dermed skrives som #5af.

I tillegg til selve fargeangivelsene, kan vi også for RGB benytte en alternativ angivelse kalt **rgba** der vi også angir en alpha. Denne siste verdien forteller hvor gjennomsiktig fargen skal være. En alpha på 0 er helt gjennomsiktig, og altså ingen synlig farge i det hele tatt. En alpha på 1 er en totalt u gjennomsiktig farge:

```
p {
    color: rgba(191,110,255,0.6);      /* Lillafarge på teksten, 60%
synlig */
}
```

De vanligste egenskapene å sette farge på i CSS er **color** (tekstfarge) og **background-color** (bakgrunnsfarge).



Forsøk selv:

- Åpne mal-filen i din kodeeditor og skriv inn følgende i <body>-delen:

```
<p id="tekst1">Farge med rgb-kode</p>
<p id="tekst2">Farge med heksadesimal kode</p>
<p id="tekst3">Farge med rgba-kode og gjennomsiktighet</p>
```

Som du ser har hver <p>-tagg fått hver sin id. Disse skal vi bruke i stilarket når vi skal teste ut ulike måter å angi farge på.

- Endre silarkreferansen til *fargerstil.css*
- Lagre med navnet *farver.html* i mappa *kapittel4*
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden under:

```
#tekst1 {
    color: rgb(191,110,255);
}

#tekst2 {
    color: #bf6eff;
}

#tekst3 {
    color: rgba(191,110,255,0.6);
}
```

5. Lagre med navnet fargerstil.css
6. Se på farger.html i valgt nettleser

Tekst

Tekstinnholdet i et element kan stilsettes ved en rekke egenskaper og tilhørende verdier. Vi vil her ta for oss noen av de vanligste.

For å gi en skråstilt stil (*kursive*) kan vi sette egenskapen **font-style** til enten **italic** eller **oblique**. Disse gir to ulike varianter av kursive, der **italic** nok er den vanligste. Dersom du ønsker å skru av **font-style**, settes verdien til **normal**.

```
p {  
    font-style: italic;  
}
```

Tekst i italic

Tekst i oblique

Vi kan også understreke, overstreke og gjennomstreke tekst ved å sette **text-decoration** til **underline**, **overline** eller **line-through**. For å fjerne linjene kan du sette **text-decoration** til **none**.

```
p {  
    text-decoration: underline;  
}
```

Tekst i underline

Tekst i overline

Tekst i line-through

Tykkelsen på skriften (fet skrift) kan vi sette gjennom egenskapen **font-weight**. Denne egenskapen har en rekke ulike verdier slik som **bold**, **bolder**, **lighter** og **normal**. Vi kan også angi tallverdier, der f.eks 400 tilsvarer normal tekst og 700 er det samme som verdien **bold**.

```
p {  
    font-weight: bolder;  
}
```

Størrelsen på teksten angis i egenskapen **font-size**. Her kan vi velge å sette den som et absolutt mål (**cm**, **px**, **pt** osv.):

```
h1 {
    font-size: 12pt;
}
```

Som en prosentangivelse i forhold til normal tekst:

```
h1 {
    font-size: 150%;
}
```

Eller som en av de ferdigdefinerte størrelsene **xx-small**, **x-small**, **small**, **large**, **x-large**, **xx-large**, **smaller**, **larger** og standardstørrelsen **medium**:

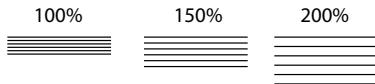
```
h1 {
    font-size: larger;
}
```



Ettersom alle de ferdigdefinerte tekststørrelsene er relative til fontstørrelsen brukeren har valgt er det svært lurt å bruke disse sett ut i fra et tilgjengelighets-hensyn. «Låser» vi font-størrelsen til f.eks. 12pt er det ikke like lett å forstørre teksten i en nettleser uten å forstørre hele nettsiden.

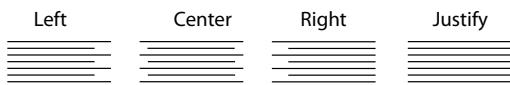
I tillegg til å justere selve tekststørrelsen kan vi også justere linjehøyden. Her kan vi også sette en absolutt verdi målt i f.eks. px eller en prosentverdi målt i forhold til normal linjehøyde. Et alternativ til å angi prosenter er å angi tall slik som 1, 2 eller 3 ganger normal forntstørrelse.

```
p {
    line-height: 250%;
}
```



Vi kan benytte egenskapen **text-align** for å fortelle hvordan vi ønsker at tekst inne i et element skal være justert. Vi kan her velge å venstrejustere (**left**), høyrejustere (**right**), midtstille (**center**) eller blokkjustere til begge marger (**justify**).

```
p {
    text-align: center;
}
```



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Skyggelegging på tekst gjøres ved hjelp av egenskapen **text-shadow** med verdier for horisontal forskyvning, vertikal forskyvning og farge:

```
h1 {  
    text-shadow: 5px 5px #faa;  
}
```

Tekst med skygge
~~Tekst med skygge~~

Vi kan også forandre hvordan tekst presenteres. Ved å benytte egenskapen **text-transform** kan vi tvinge tekst til å være **uppercase**, **lowercase** eller **capitalize** (stor forbokstav i hvert ord), selv om ikke dette er slik teksten er skrevet i HTML-dokumentet.

```
h1 {  
    text-transform: uppercase;  
}
```

Merk deg at det å sette **text-transform** til f.eks. **uppercase** ikke forandrer selve innholdet, men kun utseendet. Hadde vi endret teksten til uppercase i HTML ville vi for det første forandret meninga (vi røper) og vi ville mistet informasjon om små/store bokstaver slik at det var vanskelig å endre tilbake.



Skrifttyper

Vi endrer *skrifttype* (font) ved hjelp av egenskapen **font-family**. Dette skulle i utgangspunktet være så greit som at vi da oppgir fontnavn som verdi, slik som:

```
p {  
    font-family: "Times New Roman";  
}
```

Husk at fontnavn som består av flere ord bør ha hermetegn rundt navnet.



Dessverre er det slik at ikke alle brukere har installert alle fonter. Vi kan derfor ikke være sikker på at brukeren har installert den fonten vi angir. Har ikke brukeren det, vil en nærmest tilfeldig standardfont benyttes.

Vi vil sjeldent selv oppleve problemer med fonter, nettopp fordi vi selv har de installert. Mange får seg derfor en overraskelse når nettsider publiseres offentlig og tilbakemeldinger fra brukerne begynner å komme inn.



For å delvis fikse dette problemet bør vi også oppgi en fontfamilie og en *generisk* (felles/generell) fontfamilie når vi velger skriftype. Dersom ikke brukeren har fonten installert, vil nettleseren velge en annen font fra samme familie, og finnes ikke den heller vil det velges en tilsvarende generisk font.

```
p {  
    font-family: "Times New Roman", Georgia, Serif;  
}
```

De vanligste generiske fontene er **monospace**, **fantasy**, **serif**, **sans-serif** og **cursive**.

Selv om fontfamilier og generiske fonter fungerer som backup-løsninger, er ikke dette optimalt. Når vi har satt en font ved hjelp av egenskapen **font-family** har vi som nevnt vært helt avhengig av at brukeren som ser på nettsiden har hatt denne fonten installert på maskinen for å få opp eksakt samme font.

Utgiverne av nettlesere ble derfor enige om et sett med såkalte *Web Safe Fonts* som alle nettlesere skulle sørge for at brukerens maskin hadde installert. Hvor mange dette skal være, og eksakt hvilke fonter som er med og ikke med i listen er det imidlertid litt uenigheter om, noe som bryter med hele ideen. Dette er imidlertid de vanligste:

Arial, Helvetica, sans-serif

Arial Black, Gadget, sans-serif

Comic Sans MS, cursive, sans-serif

Courier New, Courier, monospace

Georgia, serif

Impact, Charcoal, sans-serif

Lucida Console, Monaco, monospace

Lucida Sans Unicode, Lucida Grande, sans-serif

Palatino Linotype, Book Antiqua, Palatino, serif

Tahoma, Geneva, sans-serif

Times New Roman, Times, serif

Trebuchet MS, Helvetica, sans-serif

Verdana, Geneva, sans-serif

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Som nevnt er imidlertid problemet det at man ikke er helt enig om hvilke fonter listen skal inneholde, og de fontene man er enige om blir ofte litt kjedelige og mye brukt i lengden. Derfor har man nå innført et nytt konsept i CSS der man kan referere til en fontfil (fil med informasjon om fonten) i selve nettsiden. Denne teknikken kalles for *CSS WebFonts*.

I selve CSS dokumentet angir vi absolutte eller relative referanser til de ulike fontene vi ønsker benytte ved hjelp av den spesielle blokken **@font-face**. Det er vanligst at fontfiler kommer med filendelsene *ttf*, *otf*, *svg* og *eot*.

```
@font-face {  
    font-family: Superfont;  
    src: url("superfont.ttf");  
}  
  
@font-face {  
    font-family: Superfont;  
    font-weight: bold;  
    src: url("superfont_bold.ttf");  
}
```

Deretter kan vi enkelt og greit benytte fontnavnene på vanlig måte i resten av CSS-dokumentet.:

```
p {  
    font-family: Superfont, sans-serif;  
}
```

Det er ikke nødvendig å benytte WebFonts-teknikken hver gang du skal benytte fonter. Benytt gjerne noen av de mest vanlige Web Safe Fonts-fontene for å unngå dette. Denne teknikken er derimot nyttig når vi ønsker å benytte mer spesielle fonter vi ikke er sikker på om brukeren har installert.

Husk at fonter også er beskyttet av opphavsrett på samme måte som bilder. Du kan ikke bare finne en font-fil på nett og benytte denne uten å sjekke lisensbetingelsene først.

A large, stylized, black, cursive font rendering of the lowercase letters 'a' through 'p'. The letters are arranged in two rows, with 'a' through 'h' on top and 'i' through 'p' on the bottom. The font has a thick, flowing appearance.



Forsök selv:

- Åpne mal-filen i din kodeeditor og skriv inn følgende i `<body>`-delen:

```
<p>Test på å endre font</p>
```

- Endre silarkreferansen til `fonterstil.css`
- Lagre med navnet `fonter.html` i mappa `kapittel4`
- Plasser fila `superfont.ttf` som du finner i øvingsfilene til mappa `kapittel4`
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden under:

```
@font-face {
    font-family: Superfont;
    src: url("superfont.ttf");
}

p {
    font-family: Superfont, sans-serif;
}
```

- Lagre med navnet `fonterstil.css`
- Se på `fonter.html` i valgt nettleser

CSS Bilder

Som vi tidligere har sett inkluderes bilder inn i nettsider ved hjelp av ``-taggen. Det finnes imidlertid en alternativ måte, der vi kan sette inn bilder ved hjelp av CSS. Egentlig setter vi da bakgrunnsbilde på et element, men dersom elementet ikke har noe innhold, vil det kun vise et bilde.

```
<div id="bilde">
</div>
```

```
#bilde {
    width: 200px;
    height: 200px;
    background-image: url("bilde.jpg");
}
```

Utover å sette bakgrunnsbildet, kan vi også sette en rekke egenskaper på hvordan det skal oppføres seg:

background-repeat

Dersom elementet er større enn bildet, skal det da repeteres (`repeat`), repeteres horisontalt (`repeat-x`), repeteres vertikalt (`repeat-y`) eller ikke repeteres (`no-repeat`)

background-position	En angivelse av posisjonering i horisontal og vertikal retning ved hjelp av nøkkelordene left / right / center og top / bottom / center /
background-size	Her angir vi en størrelse for bredde og høyde på bakgrunnsbildet
opacity	En tallverdi som angir fra 1 (ikke transparent) til 0 (helt usynlig) hvor gjennomsiktig bakgrunnen skal være

Det store spørsmålet blir da; når skal vi benytte HTML og når skal vi benytte CSS for å sette inn bilder?

Det enkleste svaret er at dersom bildet er en del av innholdet skal det settes inn ved hjelp av HTML, og dersom det er en del av designet skal det settes inn ved hjelp av CSS. Bildet til en nyhetsartikkel skal derfor settes inn med HTML, mens en bord mellom to elementer skal settes inn med CSS. Det er imidlertid ikke alltid like lett å avgjøre tvilstilfellene.

Med HTML får vi fordelen av at bildet markeres som en del av teksten ved kopiering, at man kan høyreklikke bildet for nedlasting og at man er sikker på at bildet blir en del av utskrift. Med CSS er det lett å bytte bildet sammen med en endring av designet på siden.

Ettersom HTML-bilder også er å anse som informasjon, er det et krav om å ha en alternativ tekst for bildet. Dette er viktig for tilgjengelighet. Noe tilsvarende finnes ikke for CSS-bilder, da de kun er design.

TIPS



EKSEMPEL - Fornøyelsespark

I dette eksemplet skal du lage en nettside med informasjon om en fornøyelsespark og gi den et attraktiv utseende ved hjelp av CSS.

1. Opprett en mappe som du gir navnet *megafunpark*. Alle filene du lager i dette eksemplet skal lagres i denne mappa
2. Prøv å finne fram til et mørkt bilde vi kan bruke i bakgrunnen og lagre det som *bakgrunn.jpg*. I vårt tilfelle har vi valgt dette bildet:



3. Start din kodeeditor og skriv inn følgende tekst:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Megafunpark</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stil.css" />
  </head>

  <body>
    <h1>Megafunpark</h1>
    <p>Megafunpark har de største og mest spennende attraksjonene!</p>

    <article id="theraptor">
      <h2>The raptor</h2>
      <p>... masse tekst her...</p>
    </article>
    <article id="dizzyrun">
      <h2>Dizzyrun</h2>
      <p>... masse tekst her...</p>
    </article>

    <footer>
      <p>&copy; Megafunpark</p>
    </footer>
  </body>
</html>
```

Der det står ...masse tekst her... kan du fylle inn med egen tekst.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

4. Lagre dokumentet i mappa med filnavnet *index.html* og sjekk ut sida i nettleseren
5. Vi skal nå lage CSS-fila. Velg **Fil > Ny** i din kodeeditor og lagre fila som *stil.css*. Skriv inn følgende kode og lagre på nytt:

```
body {
    background-color: black;
    background-image: url(bakgrunn.jpg);
    background-repeat: no-repeat;

    margin: 20px;
    color: antiquewhite;
    font-family: sans-serif;
}

h1 {
    font-size: 60pt;
    color: rgb(50, 20, 0);
    text-shadow:
        0px 0px 10px yellow,
        0px 0px 30px red;
}

h2 {
    color: orange;
    text-shadow: 0px 0px 8px red;
}
```



Legg merke til at vi angir bakgrunnsfargen slik at den blir lik bakgrunnsfargen i bildet (svart). På denne måten skjuler vi kanten på bildet. Vi har også gitt overskriften en «glødeeffekt» ved å bruke en *text-shadow* som kombinerer to ulike skygge-effekter (gul og rød farge).

6. Vi ønsker at overskriften skal ha en spesiell skrifftype som vi har funnet på Google Fonts. Google Fonts gir deg tilgang til en rekke WebFonts som du kan bruke gratis på dine egne nettsider. Etter at du har valgt en skrifftype gir Google Fonts deg en link til en CSS-fil slik at vi kan bruke skriftypene direkte i vår egen CSS-fil. Vi linker til CSS-fila i **<head>**-taggen i *index.html*. I vårt tilfelle:

```
<link href="https://fonts.googleapis.com/css?family=Fontdiner+Swanky+Luckiest+Guy" rel="stylesheet">
```

Google Fonts gir deg også kodesnuttene vi trenger for å angi skriftypen i vår egen CSS-fil. I vårt tilfelle har vi limt inn følgende kode for h1-selectoren (i *stil.css*):

```
h1 {
    font-size: 60pt;
    color: rgb(50, 20, 0);
    text-shadow:
        0px 0px 10px yellow,
        0px 0px 30px red;
    font-family: 'Fontdiner Swanky', cursive;
}
```



Mer om CSS-bokser

I kapittel 2 ble konseptet med CSS box model introdusert. Her forklarte vi hvordan alle elementer i en nettside egentlig kan sees på som en firkantet boks med innhold, indre marg, kantlinje og ytre marg.

Det er imidlertid flere egenskaper som også kan justeres som «bokser» i CSS. Vi skal her ta for oss noen av disse.

Kantlinjer

Vi har tidligere sett på at vi kan skru på kantlinjer ved å sette **border-style** til **solid** og vi kan justere tykkelsen ved hjelp av **border-width**.

Det er også andre varianter av kantlinjer. Bl.a. kan **border-style** ha ulike typer linjer slik som:

`solid`

`double`

`dotted`

`groove`

`dashed`

`ridge`

`inset`

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Vi kan også justere farge på kantlinjen gjennom egenskapen **border-color**:

```
p {  
    border-color: red;  
}
```

Uavhengig av om vi har synlige kantlinjer eller ikke kan vi også runde av hjørnene i et element ved hjelp av **border-radius**, der vi angir hvor stor radius buene skal ha.

```
p {  
    width: 100px;  
    border-radius: 10px;  
    border-style: solid;  
}
```

avrundede
hjørner



Forsøk selv:

1. Åpne mal-filen i din kodeeditor og skriv inn følgende kode i <body>-delen:

```
<h1>Meg selv</h1>  
<h2>Skole og interesser</h2>  
<p>Jeg er en elev på videregående skole, VG2 som i år tar faget IT-1</p>  
<p>I tillegg til skole bruker jeg mye tid på å trenere crossfit</p>
```



2. Endre stilarkreferansen til å være *kantlinjerstil.css*
3. Lagre med navnet *kantlinjer.html* i mappa *kapittel4*
4. Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som vist under:

```
h1 {  
    border-color: red;  
    border-style: dashed;  
}  
  
h2 {  
    border-color: blue;  
    border-style: solid;  
}  
  
p {  
    width: 300px;  
    border-radius: 10px;  
    border-style: solid;  
}
```

5. Lagre med navnet *kantlinjerstil.css*
6. Se på fila *kantlinjer.html* i valgt nettleser

Marger

Som tidligere nevnt finnes det to typer marger i CSS. De indre margene mellom innholdet og kantlinjen (kalles *padding*) og den ytre marginen utenfor kantlinjen (kalles *margin*).

Egenskapene **padding** og **margin** kan settes med alle mulige verdier, både målt med absolute og relative måleenheter. Angir vi én verdi setter vi alle fire sider samtidig.

```
h1 {
    margin: 10px;
}
```

Det finnes også en variant der vi kan justere margene hver for seg.

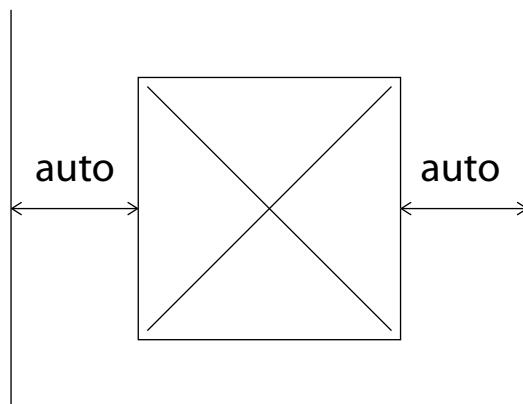
```
h1 {
    margin-top: 10px;
    margin-left: 5px;
}
```

Eller vi kan juster alle fire hver for seg, men i samme egenskap ved å angi fire verdier for henholdsvis *top*, *right*, *bottom* og *left*.

```
h1 {
    margin: 10px, 5px, 10px, 5px;
}
```

Egenskapen **margin** kan også benyttes for å sentrere elementer. Ved å sette verdien til **auto** for høyre og venstre marg, vil nettleseren fordele tilgjengelig plass likt mellom de to margene, og elementet blir altså sentrert.

```
#bilde {
    width: 300px;
    margin-left: auto;
    margin-right: auto;
}
```



Vær obs på at dersom to elementer står ved siden av hverandre vil margene legges sammen, men dersom elementene står over hverandre vil største marg benyttes som avstand.



Skygge

De fleste elementer kan få en skyggeeffekt ved å benytte egenskapen **box-shadow**. Her kan vi f.eks. fortelle at bildet skal få en skygge som går 15px mot høyre og 25px ned:

```
img {  
    box-shadow: 15px 25px;  
}
```

Som du ser vil skyggen gi en slags 3D-effekt:



Benyttes negative verdier til **box-shadow** vil dette gjøre at skyggen går motsatt vei (til venstre eller opp).



Det er også mulighet for å fortelle mer om hvordan skyggen skal se ut ved å inkludere verdier for utjevning (blur), spredning (spread) og farge:

```
img {  
    box-shadow: 15px 25px 10px 15px gray;  
}
```





Forsök selv:

- Åpne mal-filen i din kodeeditor og skriv inn de følgende kode i **<body>** delen:

```
<h1>Bilder</h1>

```

- Bildet finner du i mappen øvingsfiler, sorg for at den plasseres i mappa *kapittel4*
- Endre stilarkreferansen til å være *skyggestil.css*
- Lagre med navnet *skygge.html* i mappa *kapittel4*
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden vist under:

```
img {
    box-shadow: 15px 25px 10px 15px gray;
}
```

- Lagre med navnet *skyggestil.css*
- Se på *skygge.html* i din nettleser

Størrelse og overflow

Som vi tidligere har sett kan vi ved hjelp av egenskapene **width** og **height** sette størrelsen på et element (med noen unntak). Det er da størrelsen på innholdet vi angir, så eventuell indre marg (padding), kantlinje og ytre marg (margin) kommer i tillegg.

```
#bilde {
    width: 400px;
    height: 200px;
}
```

Merk deg stavingen på **width** og **height**. Veldig mange skriver **widht** og **height** eller **width** og **heighth**.

Hvis vi antar at padding, border og margin har lik tykkelse på alle sider blir den reelle (ytre) bredde og høyde på boksen:

Bredde = width + padding * 2 + border-size * 2 + margin * 2

Høyde = height + padding * 2 + border-size * 2 + margin * 2



Det kan ofte være kronglete å måtte beregne seg fram til dette. I CSS3 kan vi angi at **width** og **height** i stedet skal angi størrelsen til og med border (altså innhold, indre marg og kantlinje) ved å sette egenskapen **box-sizing** til å være **border-box**.

```
h1 {
  width: 200px;
  height: 50px;
  box-sizing: border-box
}
```

Elementer får vanligvis en størrelse ut i fra innholdet. Dersom vi f.eks. har en paragraf med tekst vil denne få størrelsen satt ut i fra tekstens plassbehov. Vi kan overstyre også dette ved å sette egenskapene **width** og **height**, men hva skjer med det innholdet som det ikke er plass til? Som standard vil teksten vises «på utsiden» av elementet, noe vi ser tydelig om vi setter på en kantlinje:

Lang tekst som
det nok ikke
blir plass til
inne i en liten
paragraf. Blir
det ikke plass
er det et godt
eksempel på
overflow.

Vi kan imidlertid fortelle nettleseren hva vi ønsker skal bli gjort med overskytende innhold gjennom egenskapen **overflow** eller de to mer spesialiserte egenskapene **overflow-x** og **overflow-y**.

Standard verdi er **visible**, som vi tidligere har sett at lar innhold som er for stort vises på utsiden av elementet. Verdien **hidden** vil derimot skjule det som er for mye:

Lang tekst som
det nok ikke
blir plass til
inne i en liten
paragraf. Blir
det ikke plass

Verdien **auto** vil derimot lage scroll dersom det blir for lite plass, mens verdien **scroll** vil vise scroll uansett:

Lang tekst
som det nok
ikke blir
plass til inne
i en liten
paragraf

Dersom vi ikke ønsker å sette en helt bestemt størrelse kan vi også benytte egenskapene **min-width**, **min-height**, **max-width** og **max-height**. Disse er svært kjekke for å la elementer få automatisk tilpasset en størrelse innenfor visse rammer.

```
p {
  width: 400px;
  min-height: 400px;
  max-height: 800px;
  overflow-y: auto;
}
```



Forsök selv:

- Åpne mal-filen i din kodeeditor og skriv inn følgende kode i **<body>** delen:

<p>Dette er en tekst som er lang. Hva skjer dersom den ikke får plass i boksen? Vil den falle utenfor boksen?</p>

- Endre stilarkreferansen til *overflowstil.css*
- Lagre med navnet *overflow.html* i mappa *kapittel4*
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som vist under

```
p {
    width: 200px;
    height: 50px;
    border-style: solid;
}
```

- Lagre med navnet *overflowstil.css*
- Forhåndsvise *overflow.html* i valgt nettleser. Som du ser går teksten utenfor kantlinjen
- Endre i stilarket og legg til følgende egenskap og verdi:
overflow:auto;
- Lagre stilarket og oppdater visningen av *overflow.html*

Visningsmetoder

Alle boks-elementer har også en visningsmetode. Denne settes gjennom egenskapen **display**.

Elementer som **<p>** og **<h1>** har denne som standard satt til **block**, noe som betyr at det blir et linjebrudd før og etter. Altså at elementet blir en egen «blokk». Elementer som ****, **** og **<a>** har standardverdien **inline**, noe som betyr at elementet blir en del av en tekstlinje.

Vi kan bytte visningsmetode for alle elementer. Slik at vi f.eks. kan la en link bli en egen blokk:

```
a {
    display: block;
}
```



Det finnes en rekke ulike visningsmetoder slik som `list-item`, `table`, `tablerow`, `run-in` osv. Den mest vanlige å benytte i tillegg til `inline` og `block` er imidlertid `inline-block`, altså en kombinasjon av de to. Ved å gi et element visningsmetoden `inline-block` er det fortsatt en del av linja det står på, men vi kan allikevel justere bredde og høyde, noe vi ikke kan med et element av typen `inline`.

Vi kan ikke sette egenskapene `width` og `height` på et element med egenskapen `display` satt til typen `inline` (husk at dette er standardverdi for en del elementer).



Nettleserprefikser

Den nyeste versjonen av CSS (den såkalte CSS3) gir enda flere muligheter for å bestemme utseende på nettsiden, men vær klar over at ikke alle nettlesere støtter CSS3 fullt ut etter standarden ennå. Noe egenskaper må angis spesielt for ulike nettlesere, ettersom de ikke følger egenskapene 100% slik de er definert. Dette gjøres ved å legge til et *prefiks* for ulike nettlesere.

Vi kan f.eks. lage 3 kolonner inne i alle `<div>`-tagger ved å bruke egenskapen `columns`:

```
div {  
    columns: 3;  
}
```

Denne koden vil ikke fungere i de fleste nettlesere. For å få den til å fungere i Chrome, Opera eller Safari må vi bruke `-webkit-` som prefiks. For Firefox kreves `-moz-` og Internet Explorer bruker `-ms-`. Koden som fungerer for de fleste nettlesere blir dermed:

```
div {  
    -webkit-columns: 3;  
    -moz-columns: 3;  
    -ms-columns: 3;  
    columns: 3;  
}
```

Du finner en oversikt over hvilke egenskaper i CSS3 som krever prefiks på www.w3schools.com. Av plasshensyn bruker vi kun `-webkit-` i eksemplene i boka der prefikser er påkrevd. Etterhvert som tiden går, og nettleserene utvikles, vil nettleserprefikser for CSS3 bli mindre nødvendig.



5 Mer om layout og design

I dette kapitlet vil du lære

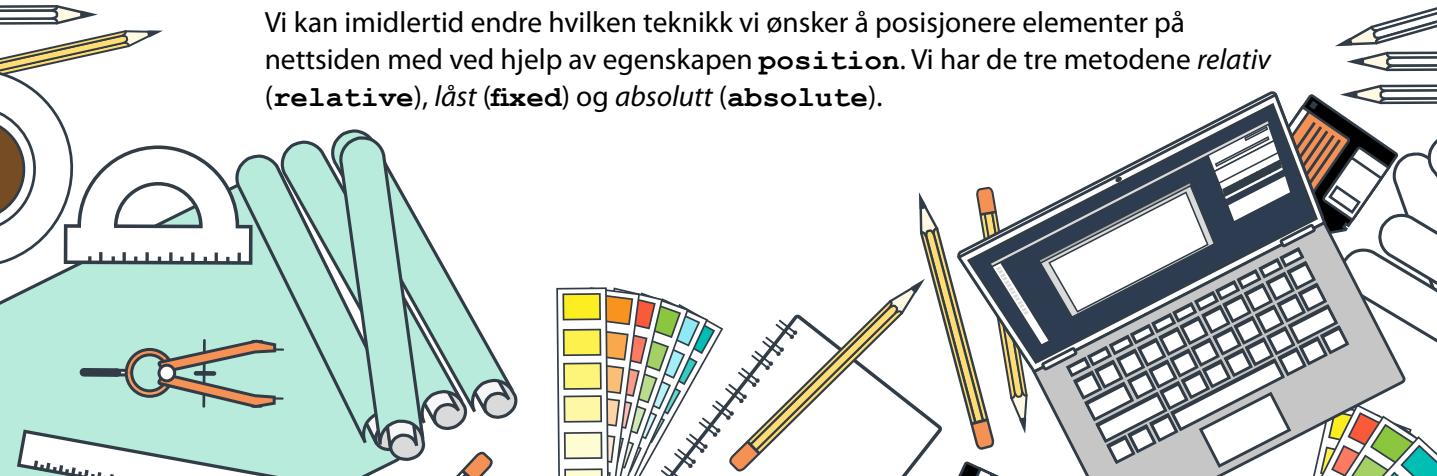
- om posisjonering
- om layout-teknikken **float**
- om layout-teknikken **flexbox**
- å sette opp nettsider med layout-teknikker
- om ulike medietyper
- om mediaqueries
- om responsive web design
- å lage menyer

I dette kapitlet skal vi ta for oss hvordan vi kan lage mer avanserte design og layout på en nettside ved hjelp av CSS. Dette kan vi gjøre gjennom ulike teknikker for posisjonering, samt teknikker for å detektere og tilpasse nettsiden til ulike medietyper og enheter.

Posisjonering

Som standard er alle elementer i HTML posisjonert ved hjelp av teknikken **static**. Dette betyr at de vises i den rekkefølgen de står i HTML-dokumentet, og at alt følger venstrekanten nedover. Elementer der **display** er satt til **block** lager nye linjeskift, mens elementer av typen **inline** kun følger på samme linje helt til det ikke lenger er mer plass, og en ny linje er nødvendig.

Vi kan imidlertid endre hvilken teknikk vi ønsker å posisjonere elementer på nettsiden med ved hjelp av egenskapen **position**. Vi har de tre metodene *relativ* (**relative**), *låst* (**fixed**) og *absolutt* (**absolute**).



Relativ posisjonering

Den enkleste er teknikken er **relative**. Med denne teknikken posisjonerer vi elementene i forhold til hvordan de ellers ville blitt plassert med **static**. Vi kan f.eks. si at alle overskrifter skal trekkes 50px lengre til høyre enn der de opprinnelig skulle vært plassert ved å sette egenskapen **left** til **50px**.

```
h1 {
  position: relative;
  left: 50px;
}
```

Overskrift 1

Tekst

Overskrift 2

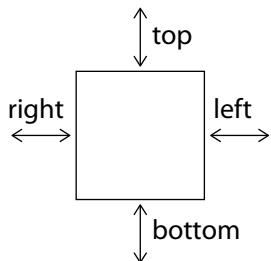
Tekst

Her har vi markert opprinnelig posisjon med en rød ramme, for å tydeligere vise forflytningen.

Det er ingenting i veien for å flytte noe delvis på «utsiden» av venstre kant eller over toppen av nettsiden. Dette er f.eks. en fin teknikk for å kun vise deler av en grafikk.

TIPS

Når vi arbeider med posisjonering kan vi justere de fire egenskapene **top**, **left**, **right** og **bottom**. Disse kan få både positive og negative verdier. Alle måleenheter er tillatt å benytte, så du kan sette både absolute måleenheter, slik som **px**, og relative måleenheter, slik som **%**.



Husk at vi kun kan justere én av egenskapene **top** og **bottom** av gangen for et element. Det samme gjelder med **left** og **right**.

MERK

En utfordring med relativ posisjonering er at den opprinnelige posisjonen til elementet fortsatt blir «holdt av» i nettsiden, og at elementet fort kan komme til å overlappe med andre elementer om vi er uheldige.

```
h1 {  
    position: relative;  
    top: 37px;  
}
```

Overskrift 1

Overskrift 2

Relativ posisjonering egner seg derfor godt til «småjusteringer» av hvordan elementer plasseres, men ikke til større endringer.



Husk at marger også er en form for posisjonering, som ofte kan være lettere å arbeide med. I stedet for å flytte elementer ved hjelp av relativ posisjonering, kan vi heller øke aktuell marg.

Forsøk selv:

- Opprett en mappe og gi den navnet *kapittel5*
- Åpne mal-filen i din kodeeditor og skriv inn følgende kode i **<body>**-delen:

```
<h1>Skolen min</h1>  
<p>Skolen ligger midt i sentrum</p>  
<h1>Interesser</h1>  
<p>Jeg er interessert i film og trening</p>
```

- Endre stilarkreferansen til *relativstil.css*
- Lagre med navnet *relativ.html* i mappa *kapittel5*
- Opprett en ny fil i kodeeditoren og skriv inn følgende CSS-kode:

```
h1 {  
    position: relative;  
    left: 50px;  
}
```

- Lagre med navnet *relativstil.css*
- Vis *relativ.html* i valgt nettleser

Låst posisjonering

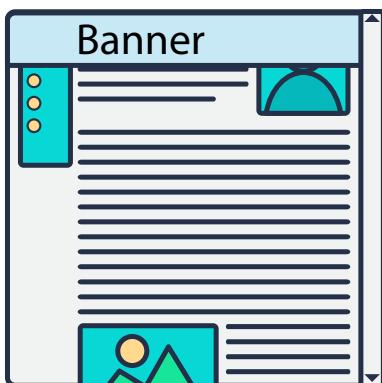
En annen posisjoneringsteknikk er **fixed**. Med **fixed** forteller vi eksakt hvor noe skal stå i forhold til øverste venstre hjørne på den delen av nettservinduet der nettsiden vises. Vi kan med andre ord plassere elementer eksakt der vi vil ha de.

På denne måten kan vi f.eks. fortelle at et bilde skal stå plassert helt i toppen av en nettside (posisjon 0 fra topp og 0 fra venstre), uten at det nødvendigvis er det første elementet i selve HTML-dokumentet:

```
#banner {  
    position:fixed;  
    top: 0;  
    left: 0;  
    width: 100%;  
}
```



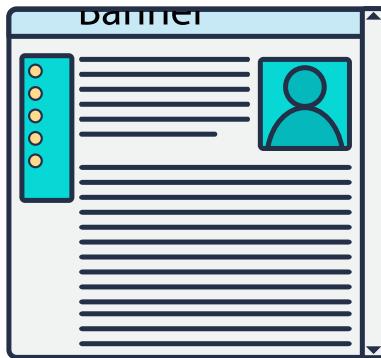
Den opprinnelige plassen blir ikke «holdt av», slik den gjør med relativ posisjonering. Det rare med posisjoneringsteknikken **fixed** er at objektet alltid plasseres i forhold til nettservinduet, og ikke i forhold til innholdet. Har vi innhold som scroller, vil elementer av typen **fixed** da «stå igjen».



Absolutt posisjonering

Den siste teknikken for posisjonering er **absolute**. Her forteller vi hvor elementet skal plasseres i forhold til elementet det er en del av.

Er elementet en direkte del av <body>-taggen (altså plassert direkte inni) vil absolute fungere mer eller mindre som **fixed**, men elementet følger med i scrollingen. Det er altså posisjonert i forhold til øvre venstre hjørne på nettsiden, og ikke øvre venstre hjørne på nettleseren.



Det spesielle med absolutt posisjonering er at vi alltid posisjonerer ut i fra elementet som ligger «rundt», eller foreldrelementet som det også kalles. Vi kan dermed f.eks. enkelt lage en tekst som dekker deler av et bilde. Det er da viktig at elementet som ligger «rundt» har en annen posisjonering enn static for at dette skal fungere.

```
<div id="bildeMedTekst">
    
    <p id="bildetekst">Bilde av giraffen vår</p>
</div>
```

```
#bildeMedTekst {
    position: relative;
    width: 300px;
}

#bilde {
    margin-left: auto;
    margin-right: auto;
}

#bildetekst {
    position: absolute;
    width: 100%;
    top: 70px;
    background-color: lightgray;
    text-align: center;
}
```



Absolutt posisjonering er derfor mest effektivt og enklest å arbeide med på «lokale» endringer.



Forsøk selv:

- Åpne mal-filen i din kodeeditor og skriv inn koden under i <body>-delen

```
<div id="bildeMedTekst">
    
    <p id="bildetekst">Bilde av frosken vår</p>
</div>
```

- Endre stilarkreferansen til *bildetekststil.css*
- Plasser bildet *froskeliten.jpg* fra øvingsfilene i mappa *kapittel5*
- Lagre med navnet *bildetekst.html* i mappa *kapittel5*
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som vist under:

```
#bildeMedTekst {
    position: relative;
    width: 300px;
}

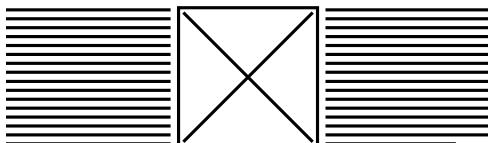
#bilde {
    margin-left: auto;
    margin-right: auto;
}

#bildetekst {
    position: absolute;
    width: 100%;
    top: 70px;
    background-color: lightgray;
    text-align: center;
}
```

- Lagre med navnet *bildetekststil.css*
- Vis *bildetekst.html* i valgt nettleser
- Kan du endre i stilarket slik at teksten blir plassert lengre opp i bildet?

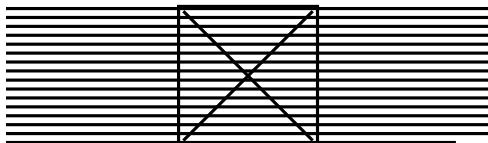
Lagjustering

En utfordring med posisjonering er at elementer lett kan komme til å overlappe hverandre.



Som regel er en slik overlapping ikke ønsket, men i noen tilfeller er det faktisk en hensikt. Vi så en slik ønsket effekt i eksempelet i forrige avsnitt der vi plasserte en paragraf oppå et bilde. Dersom vi ønsker å justere hvilken rekkefølge elementene ligger i, kan vi sette såkalt **z-index** på elementene.

Egenskapen **z-index** kan være både positive og negative verdier. Alle elementer har som standard **z-index** satt til 0, så om vi setter et element til å ha en negativ **z-index** havner det under de andre elementene. Jo lavere tall jo lengre «bakover» i nettsiden.



Tilsvarende vil en positiv verdi flytte elementer frem, og jo høyere tall jo lengre frem.

Utfordringer med posisjonering

Selv om denne typen posisjonering vi til nå har omtalt virker smart for å få plassert elementer annerledes enn der de plasseres originalt, er det mange utfordringer.

Posisjonering er derfor mindre brukt enn det man skulle tro, og man lager gjerne heller layout ved hjelp av teknikkene *floating* og *flexbox*, som vi snart skal omtale, i stedet.

Først og fremst kan posisjoneringsteknikkene bli både omfattende og forvirrende når vi beveger oss over mot større nettsider. Å posisjonere et element ender ofte med at også andre elementer rundt må posisjoneres, og til slutt vil stort sett alle deler av nettsiden være plassert med en eksakt posisjonering. Selv det å legge til litt ekstra tekst i en paragraf kan i slike tilfeller gjøre at hele designet «ryker», og at vi må beregne de fleste posisjonene på nytt.

Ofte blir også posisjonene satt ut i fra å prøve seg frem. Vi aner ikke hvorfor noe ble satt til **top: 70px** eller **right: 23%**, annet enn at det så ut til å fungere. Slike design er svært «skjøre», og noe vi bør unngå. Vi bør alltid ha en plan for hvordan designet skal være posisjonert.

En utfordring vi får når vi har utviklet et godt design som i stor grad baserer seg på posisjonering er at det sjeldent skalerer. Åpnes nettsiden med en annen skjermoppløsning eller en annen enhet er det ofte slik at ingen ting fungerer som det skal lengre. Noen anbefalinger for posisjonering kan derfor være:

- Benytt posisjonering på enkeltelementer, ikke hele nettsidene.
- Ikke benytt verdier for posisjonering som kun er funnet ved utprøving
- Planlegg posisjoneringen for alle enheter og skjermstørrelser, før du lager selve nettsiden.

Floating

Floating er en egen posisjoneringsteknikk i CSS, der vi ikke angir eksakte posisjoner, men hvordan elementer skal «flyte» i forhold til hverandre. Et enkelt eksempel på floating er om vi har et bilde og en tekst. Setter vi bildet først i paragrafen, vil det plasseres på første linje med teksten.



Loreum ipsum dolor sit amet, consectetur adipiscing elit. Nulla egestas lectus et velit luctus, eu dictum tortor luctus. Aenean at aliquet arcu. Nullam eget massa ut magna interdum viverra. Aliquam tincidunt enim sit amet magna molestie malesuada. Aliquam tempus metus eu sem vestibulum malesuada. Ut aliquam turpis massa, sit amet consequat tellus suscipit vitae. Phasellus eu dignissim velit. Ut consequat vel elit in condimentum. Donec vitae elit porttitor, varius turpis sit amet, cursus lorem. Pellentesque vel vulputate nisi. Aenean eros lacus, posuere a varius id, bibendum rutrum mauris. Nam feugiat, felis eget congue ultrices, elit metus accumsan ex, nec commodo leo magna nec enim. Proin facilisis vulputate tellus non tincidunt. Vestibulum dictum facilisis metus.



Ved å sette egenskapen **float** på bildet til **left** vil imidlertid bildet «flyte» langs venstrekanten og teksten føyer seg rundt.

```
img {
    float: left;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla egestas lectus et velit luctus, eu dictum tortor luctus. Aenean at aliquet arcu. Nullam eget massa ut magna interdum viverra. Aliquam tincidunt enim sit amet magna molestie malesuada. Aliquam tempus metus eu sem vestibulum malesuada. Ut aliquam turpis massa, sit amet consequat tellus suscipit vitae. Phasellus eu dignissim velit. Ut consequat vel elit in condimentum. Donec vitae elit porttitor, varius turpis sit amet, cursus lorem. Pellentesque vel vulputate nisi. Aenean eros lacus, posuere a varius id, bibendum rutrum mauris. Nam feugiat, felis eget congue ultrices, elit metus accumsan ex, nec commodo leo magna nec enim. Proin facilisis vulputate tellus non tincidunt. Vestibulum dictum facilisis metus.

Floating kan også benyttes for å få elementer til å følge på hverandre, og så bryte til neste linje om det blir dårlig plass. I eksempelet under har vi plassert seks ****-tagger etterhverandre i en paragraf, og gitt ****-taggene følgende CSS:

```
img {
    width: 150px;
    float: left;
}
```



Gjør vi så nettservinduet smalere, ser vi at bildene bryter ned på neste linje:



Tidligere har floating også blitt benyttet til å lage mer avanserte design, men i den nye CSS-standarden er *flexbox* et mer elegant alternativ. Vi skal derfor fokusere på *flexbox* i denne boka, som vi vil forklare i neste seksjon.

Flexbox

Flexbox er den nye og lovende posisjoneringsteknikken i CSS som også håndterer skalering av nettsiden til ulike skjermstørrelser og medietyper på en god måte. Selv om flexbox er enklere og mer stabil å arbeide med, krever den også at vi kjenner til forholdsvis mange egenskaper og konsepter. Oppsettet av HTML og CSS blir også litt mer omfattende.

For forklaringens skyld lager vi følgende struktur, der vi kun har tomme `<div>`-elementer. Elementene trenger imidlertid ikke å være av typen `<div>`, og kan selvsagt ha innhold. Merk deg spesielt at vi har en `<div>` som omslutter alle de andre elementene. Denne gis ofte `id` wrapper (henviser på det å pakke inn/samle elementene).

```
<div id="wrapper">
    <div id="boks1">Boks 1</div>
    <div id="boks2">Boks 2</div>
    <div id="boks3">Boks 3</div>
</div>
```

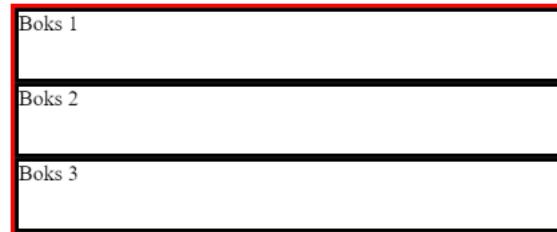
Vi setter deretter noen kantlinjer og størrelser på de ulike elementene, slik at vi enklere kan vise resultatet av CSS-kodene i denne seksjonen. Som du ser skrur vi her på rød kantlinje for elementet wrapper og sort kantlinje for den andre elementene. Dette er selvsagt ikke noen nødvendig del av flexbox-systemet.

```
#wrapper {
    border-style:solid;
    border-color:red;
}

#boks1 {
    border-style:solid;
    height:50px;
}

#boks2 {
    border-style:solid;
    height:50px;
}

#boks3 {
    border-style:solid;
    height:50px;
}
```



For så å benytte flexbox setter vi egenskapen **display** til **flex** på elementet **wrapper**. I tillegg angir vi hvordan elementene skal plasseres ut med egenskapen **flex-direction**. Her har vi valgene **row** (vannrett) og **column** (loddrett).

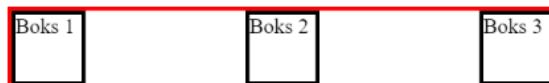
```
#wrapper {
    display: flex;
    flex-direction: row;
    border-style: solid;
    border-color: red;
}
```



Vi kan også justere avstand mellom elementene i den retningen som er valgt som **flex-direction** (hos oss **row**) ved å sette egenskapen **justify-content** til en av følgende:

center	sentrerer elementene på midten av nettsiden
flex-start	justerer elementene til venstre (ved row) eller topp (ved column)
flex-end	justerer elementene til høyre (ved row) eller bunn (ved column).
space-between	fordeler ledig plass mellom elementene
space-around	fordeler ledig plass mellom, før og etter elementene

```
#wrapper {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    border-style: solid;
    border-color: red;
}
```



En av de beste tingene ved flexbox er imidlertid at vi kan velge å gi selve elementene en variabel bredde (evt. høyde om retningen er satt til å være **column**) gjennom å si at elementet skal være **flexy**.

Ønsker vi at et element skal være flexy må vi som minimum fjerne angivelse av størrelse i retningen vi vil den skal være fleksibel i (altså **flex-direction**). I tillegg bør vi introdusere en eller flere av de tre egenskapene **flex-grow**, **flex-shrink**, **flex-basis**.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Egenskapene **flex-grow** og **flex-shrink** angir en faktor som elementene skal vokse/minke med i forhold til de andre elementene. Setter vi f.eks. **flex-grow** til 3, vil elementet vokse tre ganger så raskt som elementer der vi har satt egenskapen til 0. Egenskapen **flex-basis** forteller hvilken størrelse vi ønsker elementet skal ha før skaleringen beregnes. Merk deg at ikke alle elementene i et flexbox-system behøver være **flexy**. Vi kan fint ha noen elementer med bredde og høyde satt fast.

```
#boks1 {  
    flex-grow: 1;  
    flex-shrink: 1;  
    flex-basis: 300px;  
    border-style: solid;  
    height: 50px;  
}  
  
#boks2 {  
    border-style: solid;  
    height: 50px;  
    width: 50px;  
}  
  
#boks3 {  
    flex-grow: 3;  
    flex-shrink: 3;  
    flex-basis: 50px;  
    border-style: solid;  
    height: 50px;  
}
```



En interaktiv demo av flexbox kan du finne på nettstedet <http://flexboxin5.com/>



Det finnes også en siste egenskap kalt **flex**, som kan settes til **auto**. Dersom vi angir dette for et element vil den fylle all ledig plass, og vi trenger ikke angi noen **flex-basis**, **flex-grow** eller **flex-shrink**.

Vi kan til og med endre rekkefølgen til elementene ved å introdusere den spesielle egenskapen **order**, og så sette nummereringen vi ønsker:

```
#boks1 {
    order: 3;
    flex-grow: 1;
    flex-shrink: 1;
    flex-basis: 300px;
    border-style: solid;
    height: 50px;
}

#boks2 {
    order: 1;
    border-style: solid;
    height: 50px;
    width: 50px;
}

#boks3 {
    order: 2;
    flex-grow: 3;
    flex-shrink: 3;
    flex-basis: 50px;
    border-style: solid;
    height: 50px;
}
```



Forsök selv:

- Åpne mal-filen i din kodeeditor og skriv inn følgende kode i <body>-delen

```
<div id="wrapper">
    <div id="boks1">Boks 1</div>
    <div id="boks2">Boks 2</div>
    <div id="boks3">Boks 3</div>
</div>
```

- Endre stilarkreferansen til *flexboxstil.css*
- Lagre med navnet *flexbox.html* i mappa *kapittel5*
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som vist under

```
#wrapper {
    border-style:solid;
    border-color:red;
}

#boks1 {
    border-style:solid;
    height:50px;
}
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
#boks2 {  
    border-style:solid;  
    height:50px;  
}  
  
#boks3 {  
    border-style:solid;  
    height:50px;  
}
```

5. Lagre med navnet *flexboxstil.css*
6. Vis *flexbox.html* i valgt nettleser
7. Endre stilarkregelen for *wrapper* til

```
#wrapper {  
    display: flex;  
    flex-direction: row;  
    justify-content: space-between;  
    border-style: solid;  
    border-color: red;  
}
```

8. Lagre og vis på nytt
9. Forsøk så å endre **flex-direction** til de ulike verdiene **center**, **flex-start**, **flex-end**, **space-between** og **space-around**. Lagre mellom hver endring og se i nettleseren hva effekten blir
10. Endre stilreglene for *boks1* og *boks3* til

```
#boks1 {  
    flex-grow: 1;  
    flex-shrink: 1;  
    flex-basis: 300px;  
    border-style: solid;  
    height: 50px;  
}  
  
#boks3 {  
    flex-grow: 3;  
    flex-shrink: 3;  
    flex-basis: 50px;  
    border-style: solid;  
    height: 50px;  
}
```

11. Lagre og test i nettleseren. Forandre så nettleservinduets størrelse for å se hvordan boksene skaleres
12. Forsøk å endre verdiene for **flex-grow**, **flex-shrink** og **flex-basis** for de to boksene som har dette. Hva blir effekten når du lagrer og tester i nettleseren med størrelsесforandring på nettleservinduet?
13. Kan du også gjøre *boks2* om til en *flexy* boks?

Flexbox og layout i flere dimensjoner.

Vi kan også benytte flexbox for å få rader av elementer (alternativt kolonner av elementer) til å bryte. Dette gjøres først og fremst dersom ingen av elementene er flexy og dermed har en fast størrelse. Det kan imidlertid også gjøres dersom det ikke er plass til alle elementene med sin **flex-basis** størrelse satt.

La oss ta utgangspunkt i et tilsvarende eksempel, men der vi har 15 elementer. I stedet for å sette egenskaper på hver enkelt `<div>`, tilordner vi de klassenavnet **boks**. For å få til denne «linjebrytingen» må vi sette egenskapen **flex-wrap** til å være **wrap** (i motsetning til **nowrap** som er standard).

```
#wrapper {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
    border-style: solid;
    border-color: red;
}

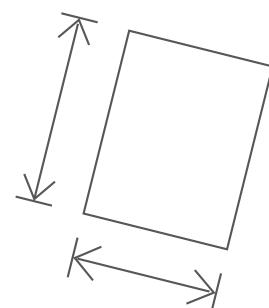
.boks {
    border-style: solid;
    height: 50px;
    width: 50px;
}
```

Boks 1	Boks 2	Boks 3	Boks 4	Boks 5	Boks 6
Boks 7	Boks 8	Boks 9	Boks 10	Boks 11	Boks 12
Boks 13	Boks 14	Boks 15			

Setter vi en høyde på `<div>`-taggen wrapper kan vi også begynne å justere hvordan elementer nå skal fordele seg i motsatt akse/retning enn det vi benytter. Egenskapen **align-content** har samme verdier som **justify-content**, men jobber da i motsatt akse av den valgte.

```
#wrapper {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: space-between;
    border-style: solid;
    border-color: red;
    height: 250px;
}
```

Boks 1	Boks 2	Boks 3	Boks 4	Boks 5	Boks 6
Boks 7	Boks 8	Boks 9	Boks 10	Boks 11	Boks 12
Boks 13	Boks 14	Boks 15			



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

I tillegg til egenskapen **align-content**, finnes også **align-items**. Også denne jobber i den andre aksen av det vi har angitt i **flex-direction**. Der **align-content** forteller noe om mellomrommets størrelse, forteller **align-items** noe om elementenes størrelse. Husk at vi også må justere **align-content** dersom vi benytter **align-item**. Vi må også fjerne elementenes størrelse i denne retningen.

```
#wrapper {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: stretch;
    border-style: solid;
    border-color: red;
    height: 250px;
}

.boks {
    border-style: solid;
    /*height: 50px;*/
    width: 50px;
}
```

Boks 1	Boks 2	Boks 3	Boks 4	Boks 5	Boks 6
Boks 7	Boks 8	Boks 9	Boks 10	Boks 11	Boks 12
Boks 13	Boks 14	Boks 15			

Dersom vi nå setter **flex-direction** til **column** kan vi se noe av fordelen med flexbox-systemet ved at vi nå omorganiserer elementene til å bli plassert ut loddrett i stedet for vannrett. Husk at vi samtidig må vi setter tilbake høyden på boksen og fjerner bredden.

```
#wrapper {
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: stretch;
    border-style: solid;
    border-color: red;
    height: 250px;
}

.boks {
    border-style: solid;
    height: 50px;
    /*width: 50px;*/
}
```

Boks 1	Boks 5	Boks 9	Boks 13
Boks 2	Boks 6	Boks 10	Boks 14
Boks 3	Boks 7	Boks 11	Boks 15
Boks 4	Boks 8	Boks 12	

En huskeregel er at alt som har med **align** å gjøre er vinkelrett på retning av valgt **flex-direction**. Alt som har med **justify** å gjøre er i samme retning som valgt **flex-direction**.



Noen vanlige layout-oppsett

Flexbox-teknikken kan også benyttes til å lage komplette sideoppsett på en forholdsvis enkel måte. De fleste nettsider vil benytte følgende seksjoner, som også er koblet opp mot de semantiske taggene:

```
<div id="wrapper">
  <header>
    Toppinnhold
  </header>
  <nav>
    Navigasjon
  </nav>
  <main>
    Hovedinnhold
  </main>
  <aside>
    Annonser
  </aside>
  <footer>
    Bunninnhold
  </footer>
</div>
```

Ved å angi en vannrett retning, ulik størrelse på elementene og **flex-wrap** kan vi lage mange ulike design. En av de vanligste er kanskje 3-kolonne-designet.



```
#wrapper {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  border-style: solid;
  border-color: red;
}

header {
  border-style: solid;
  flex-basis: 100%;
}

nav {
  border-style: solid;
  flex-basis: 200px;
}

main {
  border-style: solid;
  flex-basis: 400px;
  flex-grow: 1;
  flex-shrink: 1;
  min-height: 200px;
}

aside {
  border-style: solid;
  flex-basis: 200px;
}

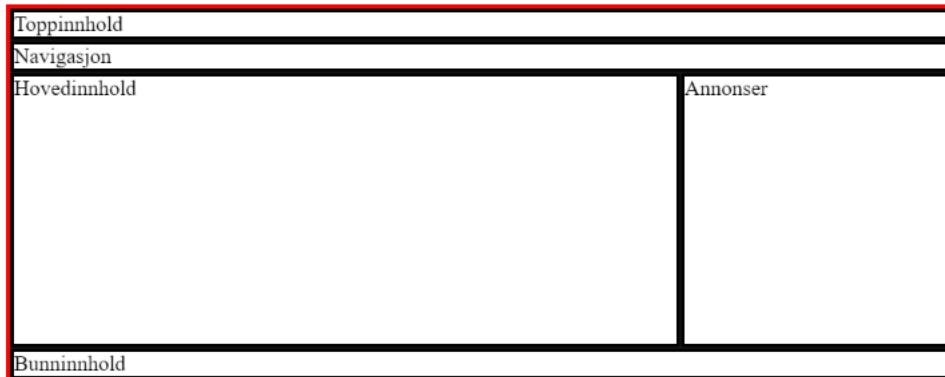
footer {
  border-style: solid;
  flex-basis: 100%;
}
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.



En liten endring i CSS-koden kan imidlertid lage et helt annet oppsett der menyen er på toppen av hovedinnholdet

```
nav {  
    border-style: solid;  
    flex-basis: 100%;  
}
```

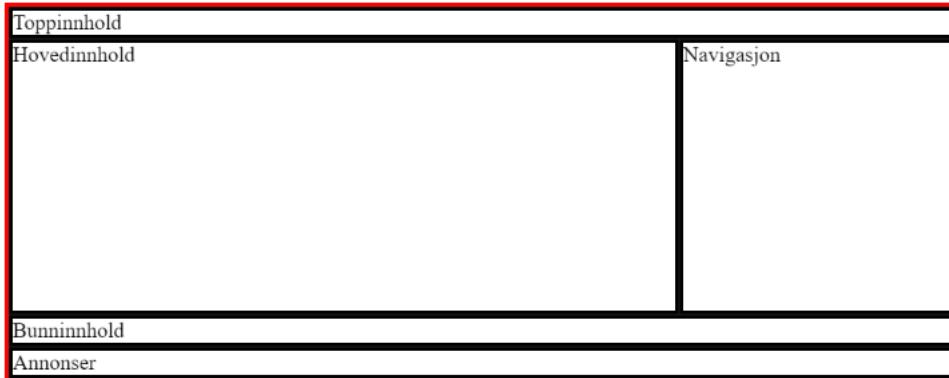


Eller vi kan totalt omrokkere på strukturen ved å benytte egenskapen **order**:

```
header {  
    order: 1;  
    border-style: solid;  
    flex-basis: 100%;  
}  
  
nav {  
    order: 3;  
    border-style: solid;  
    flex-basis: 200px;  
}  
  
main {  
    order: 2;  
    border-style: solid;  
    flex-basis: 400px;  
    flex-grow: 1;  
    flex-shrink: 1;  
    min-height: 200px;  
}
```

```
aside {
    order: 5;
    border-style: solid;
    flex-basis: 100%;
}

footer {
    order: 4;
    border-style: solid;
    flex-basis: 100%;
}
```



TIPS

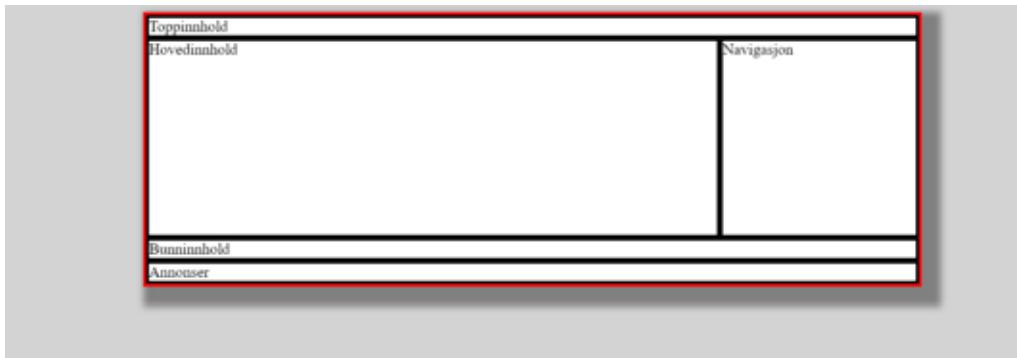
Flexbox og omstrukturering er svært nyttig sammen med *medietyper* og *medie queries* som vi presenterer senere i dette kapittelet.

Ettersom vi nå har omsluttet det som er selve «nettsiden» i en egen tagg med **id** satt til *wrapper*, kan vi også enkelt lage nettsiden smalere enn nettleservidnuet, og legge på litt bakgrunnseffekter. Dette er et vanlig design på mange nettsider.

```
body {
    background-color: lightgray;
}

#wrapper {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    border-style: solid;
    border-color: red;
    width: 800px;
    box-shadow: 10px 10px 10px 10px gray;
    background-color: white;
    margin-left: auto;
    margin-right: auto;
}
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.



Forsøk selv:

1. Åpne mal-filen og skriv inn følgende tekst i <body>-delen:

```
<div id="wrapper">
  <header>
    Toppinnhold
  </header>
  <nav>
    Navigasjon
  </nav>
  <main>
    Hovedinnhold
  </main>
  <aside>
    Annonser
  </aside>
  <footer>
    Bunninnhold
  </footer>
</div>
```

2. Endre linken til stilark slik at den referer til *designstil.cs*
3. Lagre med navnet *design.html* i mappen *kapittel5*
4. Opprett en ny fil i kodeeditoren og skriv inn CSS-koden som vist under

```
#wrapper {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  border-style: solid;
  border-color: red;
}

header {
  border-style: solid;
  flex-basis: 100%;
}

nav {
  border-style: solid;
  flex-basis: 200px;
}
```



```

main {
    border-style: solid;
    flex-basis: 400px;
    flex-grow: 1;
    flex-shrink: 1;
    min-height: 200px;
}

aside {
    border-style: solid;
    flex-basis: 200px;
}

footer {
    border-style: solid;
    flex-basis: 100%;
}

```

5. Lagre stilarket som *designstil.css* og forhåndsvise *design.html* i valgt nettleser. Endre størrelsen på nettleservinduet og se hvordan elementene endres
6. Kan du klare å fjerne annonseboksen på høyre side?

Menyer

Menyer i seg selv er ikke spesielt kompliserte. Det er jo kun en samling ordinære linker. Allikevel er det mer til det å lage en meny enn det du nok har tenkt på. For det første bør alle menyer være strukturert som en uordnet liste. Dette høres nok rart ut, for vi ønsker jo ikke kulepunkter foran menyelementene våre.

Designet kan vi imidlertid endre med CSS, som vi snart skal se. Ved å benytte en liste får vi imidlertid en struktur i menyen. Spesielt gjelder det for menyer med flere nivåer (altså menyer og undermenyer). Søkemotorer og ulike hjelpemidler for tilgjengelighet setter stor pris på denne strukturen.

Tenk deg følgende meny:

```

<nav>
    <ul>
        <li><a href="produkter.html">Produkter</a></li>
        <li><a href="bestille.html">Hvordan bestille?</a></li>
        <li><a href="kontakt.html">Kontakt oss</a></li>
    </ul>
</nav>

```

- [Produkter](#)
- [Hvordan bestille?](#)
- [Kontakt oss](#)

Det første vi kan gjøre er å fjerne kulepunkter og marger. Legg merke til at vi kun gjør dette på alle uordnede lister som er inne i en **<nav>**-tagg. Andre uordnede lister på nettsiden vil altså ikke bli påvirket.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

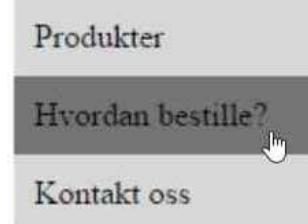
```
nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    width: 150px;
}
```

[Produkter](#)
[Hvordan bestille?](#)
[Kontakt oss](#)

Deretter kan vi stilsette listeelementene slik at vi lager en grafisk meny. Her endrer vi hvordan <a>-tagger som er inne i -tagger skal se ut. Vi endrer også fargen på bakgrunnen når en musepeker er over et element.

```
nav li a {
    display: block;
    text-decoration: none;
    background-color: lightgray;
    color: black;
    padding: 10px;
}

nav li a:hover {
    background-color: dimgray;
}
```



Om vi ønsker kan vi også stilsette menyen mer. Vi kan f.eks. legge kantlinjer rundt hele -taggen og øverst i den første og nederst i den siste -taggen:

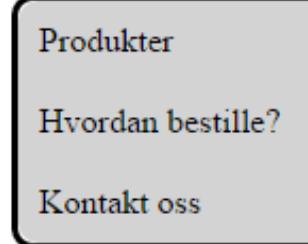
```
nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
    width: 150px;
    border-radius: 10px 10px;
    border-style: solid;
}

nav ul li:last-child a{
    border-bottom-left-radius: 10px;
    border-bottom-right-radius: 10px;
}

nav ul li:first-child a{
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
}

nav li a {
    display: block;
    text-decoration: none;
    background-color: lightgray;
    color: black;
    padding: 10px;
}

nav li a:hover {
    background-color: dimgray;
}
```



Ønsker vi i stedet å ha en horisontal meny kan vi si at listeelementene skal flyte etter hverandre. Vi må da også sette menyens bredde til å være bredere, og fylle resten av den ledige plassen med en bakgrunnsfarge. Benytter vi avrundede hjørner må vi også endre dette til å kun gjelde venstre side av første element.



```
nav ul {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    width: 100%;  
    border-radius: 10px 10px;  
    border-style: solid;  
    overflow: hidden;  
    background-color: lightgray;  
}  
  
/* nav ul li:last-child a{  
    border-bottom-left-radius: 10px;  
    border-bottom-right-radius: 10px;  
} */  
  
nav ul li:first-child a{  
    border-top-left-radius: 10px;  
    border-bottom-left-radius: 10px;  
}  
  
nav li {  
    float:left;  
}  
  
nav li a {  
    display: block;  
    text-decoration: none;  
    color:black;  
    padding: 10px;  
}  
  
nav li a:hover {  
    background-color: dimgray;  
}
```

Produkter Hvordan bestille? Kontakt oss



Forsøk selv:

- Åpne mal-filen i din kodeeditor og skriv inn koden vist under i **<body>**-delen

```
<nav>
  <ul>
    <li><a href="produkter.html">Produkter</a></li>
    <li><a href="bestille.html">Hvordan bestille?</a></li>
    <li><a href="kontakt.html">Kontakt oss</a></li>
  </ul>
</nav>
```

- Endre CSS-referansen slik at du henviser til fila *menystil.css*
- Lagre med navnet *meny.html* i mappa *kapittel5*
- Forhåndsvise *meny.html* i nettleser
- Opprett en ny fil i kodeeditoren og skriv inn CSS-koden vist under

```
nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  width: 150px;
}
```

- Lagre med navnet *menystil.css*
- Oppdater *meny.html* i nettleseren og se at kulepunktene er borte
- Legg til koden under i stilarket *menystil.css* i tillegg til det som alt er der:

```
nav li a {
  display: block;
  text-decoration: none;
  background-color: lightgray;
  color: black;
  padding: 10px;
}

nav li a:hover {
  background-color: dimgray;
}
```

- Lagre og vis *meny.html* på nytt i nettleseren



Medietyper

Til nå har vi vært vant til at en nettside kun har ett stilark koblet til seg. Det er imidlertid ikke noen begrensning som sier at en nettside ikke kan ha flere eksterne stilark:

```
<link rel="stylesheet" type="text/css" href="stilark1.css" />
<link rel="stylesheet" type="text/css" href="stilark2.css" />
```



I forbindelse med CSS benytter man ordet **spesifikk** om hvor detaljert/ innsnevret selectoren til en regel er i forhold til en annen. En id er f.eks. mer spesifikk enn **img**.

I slike tilfeller vil nettsiden benytte begge stilarkene. Skulle noen egenskaper være beskrevet i begge stilarkene vil den mest spesifikke selectoren vinne. Er de like spesifikke vil det siste stilarket ha mest å si.

Selv om det kan være nyttig å dele opp designet i flere stilark, er det først når vi gir de ulike stilarkene forskjellig betydning at denne teknikken virkelig blir nyttig. Gjennom å koble på egenskapen **media** i **<link>**-taggen kan vi fortelle nettleseren i hvilken situasjon som stilarkene skal benyttes.

```
<link rel="stylesheet" type="text/css" media="screen" href="stilark1.css" />
<link rel="stylesheet" type="text/css" media="print" href="stilark2.css" />
```

I denne koden vil stilarket *stilark1.css* benyttes dersom nettsiden vises på skjerm, mens stilarket *stilark2.css* benyttes dersom nettsiden skal skrives ut på papir. Her burde vi selvsagt også endret filnavn til noe mer fornuftig, slik som *skjerm.css* og *utskrift.css*.

Det finnes en rekke ulike medietyper som er definert i CSS, slik som **all**, **screen**, **print**, **projection**, **speech**, **braille**, **tty**, **tv**, **embossed**, **handheld** og **3d-glasses**. Det er imidlertid kun **all**, **screen** og **print** som er i vanlig bruk i dag, og som vi kan være trygge på at nettleseere støtter.

Som kanskje er logisk av navnet, vil medietypen **all** være gjeldende uansett hvilket medie vi ser nettsiden gjennom. Medietypen **all** er derfor det samme som å ikke si noe om mediotype i det hele tatt.

Medietypen **print** er, som navnet tilsier, ment for å styre utseendet på utskriften. Vi har to muligheter å lage stiler for utskrift på. Enten kan vi lage to helt separate stilark, der det ene eller det andre blir benyttet. Altså det samme som vi nettopp viste:

```
<link rel="stylesheet" type="text/css" media="screen" href="skjerm.css" />
<link rel="stylesheet" type="text/css" media="print" href="utskrift.css" />
```

Det andre alternativet er å lage ett stilark som gjelder medietypen **all**, og deretter ett stilark som inneholder unntak fra dette og som kun gjelder ved utskrift.

Ettersom medietypen **print** er mer spesialisert enn **all** vil det som er nevnt i **print** overstyre det som alt er nevnt i **all**:

```
<link rel="stylesheet" type="text/css" media="all" href="stil.css" />
<link rel="stylesheet" type="text/css" media="print" href="utskrift.css" />
```

Hvilken av de to metodene du bør velge avhenger av hvor store forskjeller det er mellom design på skjerm og utskrift. Ved små endringer er det lettest å gjøre unntak, og dermed bruke **all** og **print**. Ved store endringer er det lettest å begynne på nytt, og dermed bruke **screen** og **print**.

Husk at du ikke trenger skrive ut nettsiden på papir for å teste utskrift. Det er nok å velge forhåndsvisning i nettleseren. I eksempelvis Chrome vises forhåndsvisning automatisk når du velger skriv ut.

TIPS

Medietyper i samme stilark

Det er også en mulighet å ha flere medietyper i samme stilark, og ikke nevne noe om mediotype i `<link>`-taggen. Dette kan gjøres gjennom å lage egne `@media`-blokker. Alt som ikke står i en slik blokk vil utføres for alle medier (tilsvarende meditypen **all**), mens de mer spesifikke reglene i `@media`-blokker vil overstyre og utføres i tillegg for de aktuelle mediene.

I følgende eksempel vil alle paragrafer bli vist med rød tekst, unntatt ved meditypen **print** der de blir vist med blå tekst:

```
p {  
    color: red;  
}  
  
h1 {  
    color: green;  
}  
  
@media print {  
    p {  
        color: blue;  
    }  
}
```



Spesielle CSS funksjoner for utskrift

Flere av mediotypene i CSS har også definert sine egne egenskaper som kan være nyttige.

For utskrift er bl.a. egenskapene **page-break-inside**, **page-break-before** og **page-break-after** mye i bruk. Med disse kan vi styre hvor sideskift (nytt ark) skal komme. Mulige verdier er **auto** (standard - legg inn der det passer), **always** (alltid sideskift) og **avoid** (unngå sideskift).

Med disse kan vi f.eks si at vi ønsker å unngå sideskift midt i tabeller, men ønsker alltid et sideskift før overskrifter av typen <h2>:

```
table {
    page-break-inside: avoid;
}

h2 {
    page-break-before: always;
}
```

 Merk deg at det heter *avoid* og ikke *never*. Det er umulig å garantere at sideskift ikke vil forekomme. Dette gjelder f.eks. i tabeller som er større enn en side.

Det er også mulig å plassere inn f.eks. <div>-tagger på ønskede steder i dokumentet

```
<div class="sideskift"></div>
```

og så la de styre sideskift ved hjelp av CSS.

```
.sideskift { display:none; }

@media print {
    .sideskift {
        display:block;
        page-break-before:always;
    }
}
```

Selv innstillinger slik som anbefalt papirformat samt ekstrainformasjon om arkets marger og orientering kan styres. Vi har til og med mulighet til å endre hva som skal vises i hjørnene av arket. Alt dette gjøres gjennom den spesielle blokken @page som man legger i stilarket for utskrift.

I følgende eksempel lar vi ønsket orientering være liggende og setter margin rundt kanten av arket til å være 40px. I tillegg viser vi en logo (bilde) øverst til venstre, teksten *Sidetittel* øverst til høyre og sidenummeret nederst til venstre.

```
@page {
    size: landscape;
    margin: 40px;
    @top-left {
        content: url("logo.gif");
    }
    @top-right {
        content: "Sidetittel";
        font-size: 12px;
    }
    @bottom-right {
        content: counter(page);
    }
}
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Vi kan til og med lage ulike formateringer på forskjellige deler av utskriften ved å lage flere slike blokker med `@page : first` (første side), `@page : left` (venstresider i oppslag), `@page : right` (høyresider i oppslag) og `@page : last` (siste siden) som innledning av blokkene.

Noen gode råd for å lage utskriftsvennlige design

Det finnes ikke noen fasit for hvordan en utskriftsvennlig side skal se ut, men her er noen råd:

- Fjern unødvendig fargebruk for å spare blekk/toner.
- Vurder å fjerne bakgrunnsbilder og annet design. Ved utskrift er ofte «det enkle» best.
- Sørg for at fargebruk også passer utskrift i gråtoner. Dette gjelder f.eks. dersom ulike farger benyttes som markering, og de på sort/hvitt-utskrift ser til forveksling like ut.
- Vurder å benytte serif-fonter, da disse er ansett å være lettere å lese på papir.
- Sørg for at størrelser på skrift og bilder også er lesbare for mennesker med litt dårligere syn.
- Fjern innholdet i nettsiden som ikke er relevant ved utskrift. Vurder f.eks. om menyen og annonser skal være med.
- Fjern eller erstatt interaktivt og dynamisk innhold slik som video og animasjoner.
- Sørg for at designet ikke er bredere enn et A4-ark i bredde, og unngå dermed at du får utskrifter som er delt på flere ark både horisontalt og vertikalt.
- Benytt CSS sine muligheter for å lage naturlige sidebrudd.



Media queries

Lenge trodde man at medietyper skulle være nok til å løse alle problemer rundt det at nettsider kan sees på og dermed må tilpasses flere typer enheter. I tillegg til meditypene `all`, `print` og `screen` var man bl.a. i gang med å innføre meditypen `handheld`, som skulle benyttes for designet til nettsider på alle mobile enheter.

Problemet var imidlertid; hva kjennetegner en *mobil enhet*? Her har vi alt fra svært små skjermer som styres med tastenavigasjon til store monstre av telefoner med 6 tommers touchskjermer. Og er nettbrett da en ordinær maskin (meditypen `screen`) eller en mobil enhet?



Ganske snart innså man at det var umulig å lage én (eller flere) avgrenset mediotype for denne typen enheter. Derfor har man heller innført et nytt system som kommer i tillegg til medietyper, og som kalles *media queries*.

Med media queries kan vi selv sette regler basert på f.eks. skjermstørrelser, fargegjengivelser eller om innholdet vises på et stående eller liggende medium. Som et eksempel kan vi si at tekst skal til vanlig være sort, men dersom skjermstørrelsen er større enn 500 piksler (altså minimum 500 piksler) i bredde skal teksten være rød:

```
body {
    color: black;
}

@media all and (min-width: 500px) {
    body {
        color: red
    }
}
```

Alternativt kan vi lage egne stilark som kun blir koblet inn mot HTML-fila dersom visse betingelser er oppfylt. Dette angis da i egenskapen **media** i <link>-taggen:

```
<link rel="stylesheet" media="all and (orientation:portrait)" href="stil.staaende.css" /> 
<link rel="stylesheet" media="all and (orientation:landscape)" href="stil.liggende.css" /> 
```

Systemet bygger på at vi kan hente ut verdier for bl.a. følgende egenskaper:

width	Bredde på skjermflate
height	Høyde på skjermflate
device-width	Bredde på enheten
device-height	Høyde på enheten
orientation	Om enheten er i landscape eller portrait modus (gjelder både skjerm og papir)
aspect-ratio	Forhåndstallet i bredde/høyde for skjermflaten. Uttrykkes som f.eks. 16/9
device-aspect-ratio	Forhåndstallet i størrelse for enheten
color-index	Antall ulike farger som kan vises. f.eks. 256
monochrome	Antall gråtoner om sort/hvitt enhet. Ellers 0
resolution	Oppløsningen på enheten, slik som 300dpi

TIPS

Også **bandwidth** er foreslått som en media query-egenskap, slik at vi kan tilpasse designet ut i fra båndbredden på Internett-tilknytningen.

Det kan være vanskelig å se forskjellen på **width** og **device-width**, og det er heller ikke alltid det er noen forskjell. Årsaken til at det skiller mellom de er at noen enheter har en annen oppløsning på selve bildet (**width**) enn det den fysiske skjermen har i antall piksler (**device-width**). Det vanligste er å benytte **width** når man benytter media queries.



De fleste av media query-egenskapene som er tallverdier lar oss sette **min-** eller **max-** foran egenskapsnavnet for å sette området der regelen skal gjelde, slik som **min-width: 500px** eller **max-color-index: 256**.

Det kan også være greit å huske på at det er noen størrelser som oftere går igjen på skjermer enn andre. Det kan f.eks. være dumt å sette en medie query-regel som skiller ved eksakt 500px, da svært få enheter har denne størrelsen. De vanligste skjermstørrelsene å skille ved er:

320, 375, 414, 480, 568, 640, 667, 736, 768 og 1024 piksler.

Skjermstørrelser under 768px er ofte mobile enheter, mellom 768px og 1024px er ofte nettbrett og over 1024px er som oftest det vi litt upresist kaller «datamaskiner» (bærbar stasjonær).

Skal du benytte egenskapen **width**, kan det være lurt å legge inn følgende kode på toppen av HTML-fila, i **<head>**-seksjonen:

```
<meta name="viewport" content="width=device-width, initial-scale=1;">
```

Denne spesielle koden gjør at nettleserens bredde også blir tolket som enhetens bredde. Dermed vil vi kunne skalere nettsiden ved å endre nettleservinduet bredde.

Responsive web design

Et begrep som virkelig er i vinden nå er *responsive web design*. På tross av et forholdsvis fancy begrep og høy buzzword-status betyr dette i bunn og grunn at nettsiden skal tilpasses til ulike enheter og ulike skjermstørrelser. Med andre ord benytte teknikkene medietyper og medie queries som vi alt har jobbet litt med i dette kapittelet. Kort forklart skal sidene skaleres når størrelsen på mediet forandres, og ved visse punkter skal nettsiden også omstruktureres og få en ny layout slik at den passer bedre for den aktuelle størrelsen. Ser vi f.eks. på nettsidene til *komplett.no*, følger de tanken om responsive web design:



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Hvordan vi skal tilpasse en nettside til andre enheter (da kanskje først og fremst nettbrett og mobil) er veldig avhengig av nettstedets innhold og type. Vi kan imidlertid gi noen generelle råd. Først og fremst bør vi sørge for at nettsidens funksjonalitet er like tilgjengelig i alle mulige visningsformer. Det bør være like enkelt å handle i en nettbutikk med visning på en mobil, som på en bærbar eller stasjonær maskin. Dette høres kanskje enkelt ut, men som regel ender mobil- og nettbrett-versjonene ut med å bli «dårligere» varianter med færre muligheter og funksjoner.

For å lage gode versjoner av nettsiden selv for enheter med mindre skjermopløsning kan det være lurt å starte med å lage mobil/nettbrett-utgaven først, og så heller lage en «ordinær» utgave når disse er klare. Har vi en større og mer formell designprosess bør alle utgaver designes samtidig, før man begynne r å utvikle noen av versjonene med kode.



Er skjermen liten bør vi også fokusere på å kun ha én kolonne. Marger kan vi stort sett droppe, og siden bør være «lang» i stedet for «bred». En generell tommelfingerregel er at jo mindre skjerm jo «enktere» design bør benyttes. Unngå også å sette eksakte verdier på fontstørrelser, men benytt relative måleenheter. Alt vi skal klikke på bør være av en såpass størrelse at det er mulig å treffe med en finger (touch).

Skalere bilder

Mobiler og nettbrett har til nå hatt begrenset båndbredde og ofte kostnadsbegrensninger med tanke på overføring av data. Selv om ikke dette er et like stort problem som før, bør vi forsøke å vise bilder i en opplosning som passer skjermen. Er bilde satt inn med CSS kan vi endre hvilken URL som lastes i egenskapen background-image.

```
#bilde {  
    width: 500px;  
    background-image: url("giraff-dekstop.jpg");  
}  
  
@media all and (max-width: 768px) {  
    #bilde {  
        background-image: url("giraff-mobil.jpg");  
    }  
}  
  
@media all and (max-width: 1280px) {  
    #bilde {  
        background-image: url("giraff-nettbrett.jpg");  
    }  
}
```

Er bildene satt inn med HTML bør den nye <picture>-taggen benyttes. Denne taggen fungerer omtrent slik som <video>-taggen ved at vi angir flere ulike kildefiler. Valget av kildefil gjøres derimot ved hjelp av en medie query-betingelse.

```
<picture>  
    <source srcset="giraff-mobil.jpg" media="(max-width: 720px)">  
    <source srcset="giraff-nettbrett.jpg" media="(max-width: 1280px)">  
    <source srcset="giraff-desktop.jpg">  
      
</picture>
```

Resultatet av <**picture**>-taggen er det samme som et ordinært bilde i nettsiden, men det velges altså her blant tre ulike filer ut fra hvilken oppløsning (skjermbredde) enheten vi leser nettsiden på har.

Er enheten under 720px i bredde velges *giraff-mobil.jpg* som har en liten filstørrelse, og dårligere kvalitet. Er enheten under 1280 velges i stedet *giraff-nettbrett.jpg* som har litt større filstørrelse, og da også litt bedre kvalitet. I alle andre tilfeller velges *giraff-desktop.jpg* som har stor filstørrelse og svært god kvalitet.

Skulle ikke <**picture**>-taggen være støttet av nettleseren vises heller «innholdet» i taggen, som jo er en ordinær <**img**>-tagg.

 Husk at om du kun skalerer ned et bildes visning ved å sette en størrelse på bildet i HTML/CSS, overføres hele bildefila allikevel. Derfor må vi benytte en mekanisme som velger blant ulike filer.

 Når vi arbeider med flere filer som er varianter av samme bilde, må vi passe på å oppdatere alle de ulike filene dersom vi ønsker endre bildet.

EKSEMPEL - Responsive design

I dette eksemplet skal vi lage en nettside som har forskjellig layout for en vanlig PC-skjerm, nettbrett og smarttelefoner.

1. Vi tar utgangspunkt i nettsiden for fornøyelsesparken du lagde i slutten av kapittel 4 og legger til en meny (<**nav**>-tag med en samling linker) før artiklene. Vi har ikke behov for at linkene skal virke. Vi angir derfor bare et firkanttegn som **href**. Til slutt lager vi en <**div**>-tag som omslutter både <**nav**>-taggen og <**article**>-taggene. Vi skal bruke denne <**div**>-taggen for å samle innholdet som flexbokser. Vi setter også inn metataggen for skalering i **head**-delen:

```
<head>
  <title>Megafunpark</title>
  <meta charset="utf-8" />
  <link rel="stylesheet" type="text/css" href="stil.css" />
  <link href="https://fonts.googleapis.com/css?family=Fontdiner+
    Swanky|Luckiest+Guy" rel="stylesheet">
  <meta name="viewport" content="width=device-width,
    initial-scale=1">
</head>
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
<body>
    <h1>Megafunpark</h1>
    <p>Megafunpark har de største og mest spennende attraksjonene!</p>

    <div id="wrapper">
        <nav id="meny">
            <ul>
                <li><a href="#">Priser</a></li>
                <li><a href="#">Åpningstider</a></li>
                <li><a href="#">Finn fram</a></li>
                <li><a href="#">Om oss</a></li>
            </ul>
        </nav>

        <article id="theraptor">
            <h2>The raptor</h2>
            <p>... masse tekst her...</p>
        </article>
        <article id="dizzyrun">
            <h2>Dizzyrun</h2>
            <p>... masse tekst her...</p>
        </article>
    </div>

    <footer>
        <p>© Megafunpark</p>
    </footer>
</body>
```

2. Denne layouten passer fint for små skjermer (for eks. smarttelefoner). Her ønsker vi som regel at alt innholdet vises under hverandre i én kolonne. Det vi ønsker å forandre på er at menyen skal vises horisontalt (`display: inline`) og uten kulepunktene i listeelementene. Vi gjør også fargen på linkene litt mer synlig samt legger til litt mer luft mellom elementene (padding). Legg til følgende kode nederst i CSS-fila ():

```
a {
    color: orange;
}

ul {
    list-style: none;
    margin: 0;
    padding: 0;
}

li {
    display: inline;
    padding: 5px;
}
```



3. Etter hvert som vi lager layout for nettbrett og PC kommer vi til å oppgi størrelse på ulike elementer. I utgangspunktet angis størrelsene for selve innholdet (et tenkt rektangel som akkurat omslutter teksten eller bildet). Vanligvis er det mer praktisk å oppgi størrelsen inkludert tykkelsen på eventuelle kantlinjer + luft mellom innholdet og kantlinjer (padding). For å gjøre dette legger vi til følgende kode øverst i CSS-fila:

```
* {
    box-sizing: border-box;
}
```

4. Vi begynner nå med å lage layout for nettbrett – dvs. skjermer som har en bredde på 768 piksler eller mer. Her ønsker vi at artiklene skal ligge som to kolonner. Legg til følgende kode nederst i CSS-fila:

```
@media all and (min-width: 768px) {
    #wrapper {
        display: flex;
        flex-wrap: wrap;
    }

    #meny {
        width: 100%;
    }

    article {
        flex-basis: 50%;
        padding-right: 8px;
    }
}
```

Her lar vi menyen få en bredde på 100% og slår på linjebryting (wrap). Artiklene blir derfor skjøvet under menyen. Vi har gitt artiklene en bredde på 50% slik at de fordeler seg i to kolonner:



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

5. Til slutt skal vi lage layout for skjermer som har en bredde på 1024 piksler eller mer. Her ønsker vi at menyen skal ligge som en egen kolonne på venstre side. Vi legger til følgende kode nederst i CSS-fila:

```
@media all and (min-width: 1024px) {

    #wrapper {
        display: flex;
    }

    #meny {
        width: 12%;
    }

    article {
        flex-basis: 44%;
        padding-right: 8px;
    }

    li {
        display: block;
    }
}
```

Her fordeler vi innholdet i bredden slik at menyen opptar 12 %, mens de to artiklene tar 44 % hver. Vi setter også block-display på -elementene slik at de vises under hverandre.



7. Test ut nettsida i nettleseren. Forandre størrelse på nettleseren og sjekk ut de tre forskjellige layoutene.

Legg merke til at overskriften er for stor til å få plass når bredden på skjermen blir svært liten. Vi kan løse dette ved å angi ulik font-størrelse i de tre forskjellige layoutene. Hvis vi i stedet ønsker å la en tekst gradvis skaleres ut i fra størrelsen på nettleseren, kan vi la teksten være et bilde og sette størrelsen til en prosentverdi.



6 Eksempel - Festival



EKSEMPEL – Festival

I dette kapittelet skal vi gjennomføre et større eksempel hvor vi lager et nettsted for en festival.

Eksemplet tar utgangspunkt i en fiktiv festival kalt *chill-out* festivalen, men du kan selvsagt endre dette til din egen favoritt-festival, for eks. en musikk-festival. I eksemplet lager vi kun to nettsider – en hovedside (*index.html*) i tillegg til en side med oversikt over festivalprogrammet (*program.html*). Du kan selv legge inn mer tekst og bilder, lage flere nettsider eller endre farger, skrifttype etc.

Nettstedet skal ha responsive design og i utgangspunktet være tilpasset små skjermer.

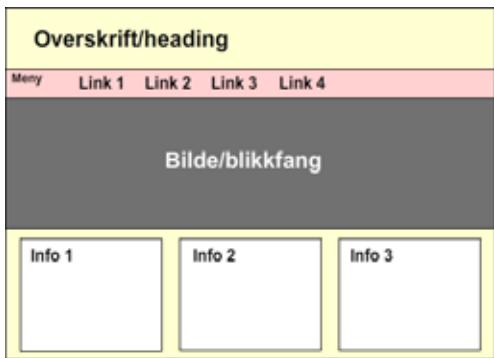
Menyen skal være felles for alle sidene på nettstedet. For små skjermer skal den være skjult i en knapp i det øvre høyre hjørnet. Når brukeren klikker på knappen skal menyen dukke opp. Menyen skal også ha en lukke-knapp slik at brukeren kan lukke menyen uten å klikke på en link.

Innholdet nederst er fordelt i én kolonne slik at alt ligger under hverandre:



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

For større skjermer skal vi ha layouten vist nedenfor. Legg merke til at menyen vises som en egen rad ovenfor bildet, samt at innholdet nederst er fordelt på tre kolonner:



Før du begynner trenger vi noen bilder som vi bruker som knapper for å åpne og lukke menyen:



aapnemeny.png



lukkmeny.png

Du kan enten lage dem selv, eller hente dem ned fra eksempelfilene. I vårt tilfelle har bildene en størrelse på 32 x 32 piksler. Du trenger også et bilde som skal virke som blikkfang. Vi har brukt et bilde som vi har kalt *kveld.jpg*:



Lage menyen

- Opprett en mappe inne i webutviklings-mappa som du kaller *kapittel6*. Lagre alle filer til dette prosjektet i denne mappa
- Vi begynner med å lage menyen. Ta utgangspunkt i malen, lagre den som *index.html* og endre den slik at vi får denne koden:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Chill-out festivalen</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="nettstedstil.css" />
    <meta name="viewport" content="width=device-width,initial-scale=1">
  </head>

  <body>

    <nav id="menyen">
      <ul>
        <li><a href="program.html">Program</a></li>
        <li><a href="#">Billetter</a></li>
        <li><a href="#">Festivalområdet</a></li>
        <li><a href="#">Overnatting</a></li>
        <li><a href="#">Transport</a></li>
        <li><a href="#">Mat og drikke</a></li>
        <li><a href="#">Kontaktinfo</a></li>
      </ul>
    </nav>

  </body>
</html>
```

Legg merke til at vi har lagd en **<nav>**-tag som inneholder en meny (****-liste som inneholder linker). Foreløpig linker vi bare til programsiden (som vi skal lage senere). Vi har derfor bare angitt # (internlink til ingenting) som **href** for de andre linkene. Nedenfor ser du hvordan siden vises i nettleseren:



- Vi har angitt en **id** i **<nav>**-taggen som heter menyen. Meningen er å lage CSS-kode for denne taggen slik at meny-elementene ser ut som knapper. I første omgang lager vi menyen slik at den ser bra ut på små skjermer. Lag en ny fil du kaller *nettstedstil.css* og lagre den i samme mappe som HTML-fila. Skriv inn koden på neste side:

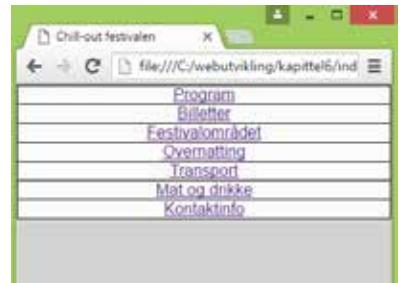
KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
* {
    margin: 0px;
    padding: 0px;
    box-sizing: border-box;
}

body {
    background-color: lightgrey;
    font-family: sans-serif;
}

#menyen ul {
    list-style: none;
    display: flex;
    flex-wrap: wrap;
}

#menyen ul>li {
    width: 100%;
    text-align: center;
    border: solid 1px black;
    margin-bottom: -1px;
    background-color: white;
}
```



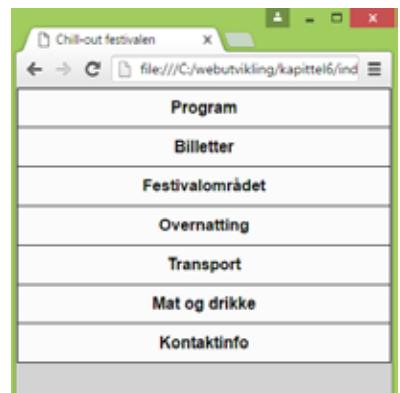
Her setter vi først størrelsen på marger og padding til 0 for alle (*) elementer. På denne måten har vi bedre kontroll og kan selv styre disse verdiene. Vi ønsker også å angi størrelser inkludert padding og kantlinjer slik at vi setter **box-sizing** til **border-box**.

Vi lar lista være en wrapper for flex-bokser ved å sette egenskapen **display** til **flex**. På denne måten kan vi lettere styre hvordan meny-elementene skal vises (vertikalt eller horisontalt).

Vi ønsker at menyen skal dekke hele bredden på skjermen. Vi setter derfor bredden til 100% på ****-taggene. Vi angir også at innholdet (**<a>**-taggene) skal ligge i senter og lager kantlinjer. Ved å sette **margin-bottom** til -1px forskyver vi elementene slik at vi unngår doble kantlinjer.

- Forhåndsvise *index.html* i nettleseren. Se at menyen hele tiden holder seg midstilt når du endrer størrelsen på vinduet i nettleseren. Legg også merke til at hver knapp bare er aktiv over teksten (**<a>**-taggene). Vi ønsker at hele knappen skal være aktiv. Vi ønsker også å justere utseende på **<a>**-taggene ved å ta bort understrekingen på teksten osv. Legg til følgende kode nederst i CSS-fila (*nuttstedstil.css*):

```
#menyen ul>li>a {
    display: block;
    padding: 10px;
    text-decoration: none;
    font-weight: bold;
    color: black;
}
```

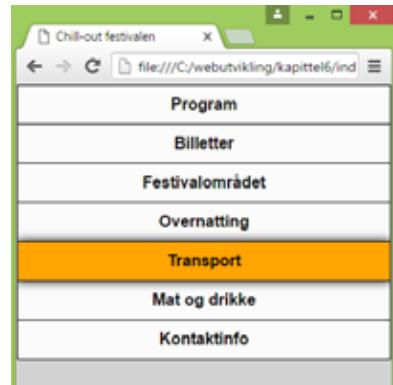


Her setter vi **display** til **block** på **<a>**-taggene slik at de fyller hele bredden. Vi øker også *padding* slik at det blir litt mer luft rundt teksten og knappene blir større. Ved å sette *text-decoration* til *none* tar vi bort understrekingen.

5. Vi vil at knappene skal framheves når vi flytter musa over dem. Legg til følgende kode i CSS-fila (*nettstedstil.css*):

```
#menyen ul>li:hover {
    box-shadow: 0px 0px 15px black;
    z-index: 1;
    background-color: orange;
}
```

Her bruker vi pseudo-klassen **:hover** for å få knappene til å reagere med musa. Når musa er over en knapp vises det en svart skygge under den. Vi setter **z-index** til 1 for å flytte knappen til toppen. På denne måten er vi sikker på at skyggen havner oppå de andre knappene (de andre knappene vil ha **z-index** lik 0):



6. Når *index.html* åpnes første gang skal ikke menyen vises - og man må klikke på bildet *aapnemeny.png* for å få vist menyen. Når menyen vises skal den kunne skjules igjen ved å klikke på bildet *lukkmeny.png*. Først må vi endre litt i HTML-koden. Vi lager to **<div>**-tagger inne i **<nav>**-taggen. Den ene **<div>**-taggen skal vise den lukkede menyen, mens den andre skal vise den åpne menyen. Inne i den ene **<div>**-taggen linker vi til den andre og omvendt (med **<a>**-tagger som inneholder åpne- og lukke-bilde):

```
<body>

<nav id="menyen">

    <div id="meny_lukket">
        <a href="#meny_aapen"></a>
    </div>
    <div id="meny_aapen">
        <a href="#meny_lukket"></a>
        <ul>
            <li><a href="program.html">Program</a></li>
            <li><a href="#">Billetter</a></li>
            <li><a href="#">Festivalområdet</a></li>
            <li><a href="#">Overnatting</a></li>
            <li><a href="#">Transport</a></li>
            <li><a href="#">Mat og drikke</a></li>
            <li><a href="#">Kontaktinfo</a></li>
        </ul>
    </div>
</nav>

</body>
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

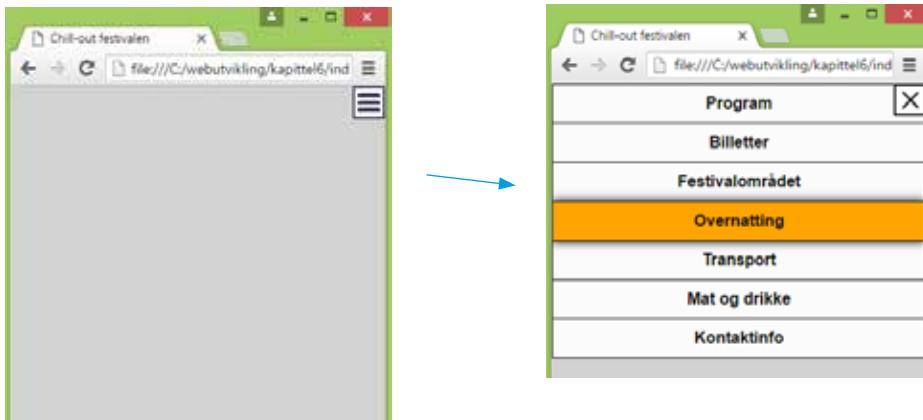
7. Vi legger til følgende kode i CSS-fila (*nettstedstil.css*):

```
#menyen div>a>img {
    position: absolute;
    right: 0px;
    z-index: 2;
}

#meny_aapen {
    display: none;
}

#meny_aapen:target {
    display: initial;
}
```

Her flytter vi først åpne/lukke bildene helt inntil høyre kant på skjermen og setter **z-index** til 2 slik at de vises over både menyen og annet innhold. Deretter skjuler vi **<div>**-taggen som inneholder den åpne menyen. Til slutt bruker vi **:target** for å vise menyen igjen når vi klikker på åpne-linken (åpne-bildet). Sjekk at du kan åpne og lukke menyen ved å klikke på åpne- og lukke-knappen:



En liten ulempe ved å lage menyen på denne måten er at menyen fortsatt vil være åpen når vi kommer tilbake fra en side (ved å bruke tilbake-knappen i nettleseren).

Grunnen er at nettleseren husker at vi hoppet fra internlinken `#meny_aapen` og går tilbake dit når vi klikker på tilbake-knappen. Hvis du ikke ønsker at menyen skal oppføre seg på denne måten kan vi lage en liten «lureløsning» med *JavaScript*.

Legg inn denne koden i HTML-fila (*index.html*) på slutten inne i **<body>**-taggen:

```
<script>window.location.href = "#";</script>
```

Javascript-koden fungerer ved at vi automatisk hopper til internlinken # (ingenting) når siden blir lastet inn i nettleseren.



7. Som du husker skulle meny-elementene stå ved siden av hverandre for større skjermer. Vi gjør dette ved å lage en *media query* for skjermbredde på 1024 piksler eller større. Legg til følgende kode nederst i CSS-fila (*nettstedstil.css*):

```
@media all and (min-width: 1024px) {

    #meny_lukket {
        display: none;
    }

    #menyen div>a>img {
        display: none;
    }

    #meny_aapen {
        display: initial;
    }

    #menyen ul {
        flex-wrap: nowrap;
        position: initial;
        justify-content: space-between;
    }

    #menyen ul>li {
        flex: auto;
        margin-bottom: 0px;
    }

    #menyen ul>li>a {
        padding: 5px;
    }
}
```

Her begynner vi med å skjule den lukkede menyen og åpne/lukke-bildene. Deretter angir vi normal (*initial*) visning på menyen. Ved å angi **nowrap** vil meny-knappene legge seg horisontalt. Vi setter **justify-content** til **space-between** slik at menyelementene fyller hele bredden på skjermen. Vi setter også **flex** til **auto** slik at det ikke blir noe luft mellom elementene. Til slutt setter vi tilbake bunn-margen til 0 piksler og gjør menyen litt smalere ved å redusere padding på **<a>**-taggene.

Endre bredden på netleseren og sjekk ut de to forskjellige layoutene:



Lage innhold

Vi skal nå legge inn resten av innholdet på nettsida.

1. Endre HTML-fila (*indeks.html*) slik som vist på neste side:

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Chill-out festivalen</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="nettstedstil.css" />
        <link rel="stylesheet" type="text/css" href="hovedsidesstil.css" />
        <meta name="viewport" content="width=device-width,initial-scale=1">
    </head>

    <body>

        <div id="wrapper">

            <nav id="menyen">
                <div id="meny_lukket">
                    <a href="#meny_aapen"></a>
                </div>
                <div id="meny_aapen">
                    <a href="#meny_lukket"></a>
                    <ul>
                        <li><a href="program.html">Program</a></li>
                        <li><a href="#">Billetter</a></li>
                        <li><a href="#">Festivalområdet</a></li>
                        <li><a href="#">Overnatting</a></li>
                        <li><a href="#">Transport</a></li>
                        <li><a href="#">Mat og drikke</a></li>
                        <li><a href="#">Kontaktinfo</a></li>
                    </ul>
                </div>
            </nav>

            <header id="tekstheader">
                <h1>Chill-out festivalen</h1>
            </header>

            <header id="bildeheader">
                
            </header>

            <main>
                <article>
                    <h2>Eventyrlige konserter!</h2>
                    <p>...masse stoff her...</p>
                </article>
                <article>
                    <h2>Flott natur!</h2>
                    <p>...masse stoff her...</p>
                </article>
                <article>
                    <h2>Inspirerende foredrag!</h2>
                    <p>...masse stoff her...</p>
                </article>
            </main>
        </div>

    </body>

</html>
```

Vi ønsker å lage layout med flex-bokser slik at vi omslutter alt innholdet i en `<div>`-tag med id lik `wrapper`. Inne i denne taggen ligger menyen, en `<header>`-tag med overskrift, en annen `<header>`-tag med et bilde og til slutt kommer hovedinnholdet (tre artikler) inne i en `<main>`-tag. Der det står ...masse stoff her... kan du legge inn egen tekst.

Grunnen til at vi har skilt overskriften og bildet i to forskjellige `<header>`-tagger er at det blir enklere å legge menyen mellom dem i layouten for større skjermer.

Legg merke til at vi har linket til en ny CSS-fil inne i `<head>`-taggen. Vi skal skrive resten av CSS-koden for hovedsida (`index.html`) i denne fila. På denne måten kan vi angi stilregler som er spesielle for denne siden - og som ikke påvirker andre sider i nettstedet.

2. Lag en ny fil du kaller `hovedsidesstil.css` og lagre den i samme mappe som HTML-fila. Skriv inn følgende kode:

```
#wrapper {
    display: flex;
    flex-wrap: wrap;
}

nav {
    width: 100%;
}

header {
    width: 100%;
    background-color: black;
    text-align: center;
}

header h1 {
    padding: 20px;
    color: white;
    text-shadow: 0px 0px 15px red;
}

header img {
    width: 100%;
    max-width: 800px;
}

main {
    width: 100%;
}
```



Her har vi redusert størrelsen på nettservinduet slik at innholdet kommer under hverandre.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Her setter vi **display** til **flex** slik at vi kan lage layout med flex-bokser. Vi slår på «linjebrytning» (**wrap**) og setter bredden på de ulike elementene til 100%. På denne måten havner elementene under hverandre. Vi forandrer også litt på farger og setter en egen stil for overskriften inne i **<header>**-taggen slik at vi får litt luft rundt og en glødeeffekt (**text-shadow**). Vi begrenser også bildet slik at det ikke kan skaleres til å bli større enn 800 piksler bredt (det bør uansett ikke skaleres mer enn den originale pikselstørrelsen på bildet). Åpne *index.html* og erstatt ...masse stoff her ... med tekst. Forhåndsvise *index.html* og forsøk ulike størrelser i nettleservinduet.

3. Vi fortsetter med å lage en bedre layout for innholdet i **<main>**-taggen. Endre koden for main (i *hovedsidesstil.css*):

```
main {
    width: 100%;
    display: flex;
    flex-wrap: wrap;
}
```

Legg også til følgende kode i CSS-fila (*hovedsidesstil.css*):

```
main article {
    width: 100%;
    padding: 10px;
    border-bottom: solid 1px grey;
}

main h2 {
    padding-bottom: 5px;
}
```

Her setter vi display til flex for **<main>**-taggen slik at vi kan lage layout med flex-bokser (inne i **<main>**). Vi lager også litt mer luft rundt artiklene og **<h2>**-overskriftene. Artiklene har også fått en kantlinje slik at de skiller mer fra hverandre.

4. Når skjermen er 1024 piksler eller bredere har vi denne layouten:



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Vi ønsker å endre layouten slik at menyen vises mellom de to <header>-taggene (overskriften og bildet). I tillegg ønsker vi at artiklene skal vises ved siden av hverandre i tre kolonner. Legg til følgende kode i CSS-fila (*hovedsidesstil.css*):

```
#menyen {  
    order: 1;  
}  
  
#tekstheader {  
    order: 0;  
}  
  
#bildeheader {  
    order: 2;  
}  
  
main {  
    order: 3;  
    flex-wrap: nowrap;  
}  
  
main article {  
    padding: 10px;  
    border: none;  
}  
}
```

Her endrer vi rekkefølgen på flex-boksene slik at menyen kommer mellom overskriften og bildet. Artiklene legger seg ved siden av hverandre ettersom vi tar bort linjebrytningen (`nowrap`) i `<main>`. Til slutt tar vi bort kantlinjene mellom artiklene:



Lage programsida

Vi skal lage en side med programmet for festivalen.

1. Ta utgangspunkt i *index.html* og lagre den som *program.html*. Slett alt inne i **<body>**-taggen, bortsett fra **<nav>**-taggen (se koden nedenfor) og erstatt *hovedsidesstil.css* med *programstil.css*. Menyvalget *Program* skal erstattes med menyvalget *Hjem* som linker til *index.html*. Det skal også vises et bilde av et lite hus ved siden av menyvalget. Sørg for å lagre øvingsfilen *hjem.png* i prosjekt-mappen før du går videre:



```
<!DOCTYPE html>
<html>
    <head>
        <title>Chill-out festivalen - program</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="nettstedstil.css" />
        <link rel="stylesheet" type="text/css" href="programstil.css" />
        <meta name="viewport" content="width=device-width,initial-scale=1">
    </head>

    <body>

        <nav id="menyen" class="meny">

            <div id="meny_lukket">
                <a href="#meny_aapen"></a>
            </div>
            <div id="meny_aapen">
                <a href="#meny_lukket"></a>
                <ul>
                    <li><a href="index.html">
                        Hjem
                    </a></li>
                    <li><a href="#">Billetter</a></li>
                    <li><a href="#">Festivalområdet</a></li>
                    <li><a href="#">Overnatting</a></li>
                    <li><a href="#">Transport</a></li>
                    <li><a href="#">Mat og drikke</a></li>
                    <li><a href="#">Kontaktinfo</a></li>
                </ul>
            </div>
        </nav>

    </body>
</html>
```

2. Vi skal lage en heading samt en liten meny (*knapper*) for dagene i festivalen. Knappene har internlinker som vil gjøre det lettere for brukeren å hoppe til ønsket dag. Legg til følgende kode etter `</nav>`-taggen i HTML-fila (*program.html*):



```

<header>
    <h1>Chill-out festivalen</h1>
    <h2>Program</h2>
</header>
<nav id="dagmeny">
    <ul>
        <li><a href="#dag1">22/6</a></li>
        <li><a href="#dag2">23/6</a></li>
    </ul>
</nav>

```

3. Vi skal lage CSS-fila for den nye nettsida. Lag en fil du kaller *programstil.css* og skriv koden nedenfor. Lagre fila i samme mappe som de andre filene:

```

body {
    background-color: black;
    color: white;
}

header {
    text-align: center;
    text-shadow: 0 0 10px yellow;
    padding: 10px;
}

#dagmeny ul {
    list-style: none;
    display: flex;
    justify-content: space-between;
}

#dagmeny ul>li {
    flex: auto;
    height: 50pt;
    text-align: center;
    border: solid 3px orange;
    border-radius: 20px;
    background-color: rgb(50,15,15);
}

#dagmeny ul>li>a {
    text-decoration: none;
    color: white;
    font-size: 40pt;
    display: block;
    width: 100%;
}

#dagmeny ul>li:hover {
    background-color: darkorange;
    box-shadow: 0px 0px 15px red;
}

```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Her begynner vi med å sette bakgrunnfargen til svart og tekstfargen til hvit. Vi overstyrer dermed bakgrunnsfargen i *nettstedstil.css*. Vær oppmerksom på at dette bare fungerer når vi skriver **<link>**-taggen til *programstil.css* etter **<link>**-taggen til *nettstedstil.css* i HTML-koden.

Vi lar ****-taggen ha flex slik at vi enkelt kan sette inn nye knapper (flere dager) som fordeler seg jevnt i bredden.

Vi tar bort understrekingen i **<a>**-taggene, setter display til block og bredden til 100%. På denne måten dekker de hele bredden inne i ****-taggene.

Til slutt bruker vi pseudo-klassen **:hover** for å endre farge og «glød» når vi beveger musa over knappene:



3. Vi fortsetter med å lage hovedinnholdet i HTML-fila. Sørg for at bildene *kaffe.jpg* og *fjelltur.jpg* er lagret i prosjektmappa. Skriv inn følgende kode etter den siste **</nav>**-taggen (i *program.html*):

```
<main>
    <div id="dag1" class="dag">
        <h1>22/6</h1>
        <div class="tid">
            <h1>kl 1700</h1>
            <h2>Overskrift</h2>
            
            <p>...mer tekst her...</p>
        </div>
        <div class="tid">
            <h1>kl 2100</h1>
            <h2>Overskrift</h2>
            
            <p>...mer tekst her...</p>
        </div>
    </div>
    <div id="dag2" class="dag">
        <h1>23/6</h1>
        <div class="tid">
            <h1>kl 1700</h1>
            <h2>Overskrift</h2>
            
            <p>...mer tekst her...</p>
        </div>
    </div>
</main>
```



Her organiserer vi dagene i inne i **<div>**-tagger. Inne i disse igjen har vi **<div>**-tagger for ulike tidspunkt. Legg til flere dager, tidspunkter, mer info, bilder, lyd, video etc. etter eget ønske.

4. Vi ønsker å endre layouten slik at de ulike dager og tidspunkt blir litt mer oversiktlig. Legg til følgende kode nederst i CSS-fila (*programstil.css*):



```
main {
    display: flex;
    flex-wrap: wrap;
}

.dag {
    width: 100%;
    padding-top: 15px;
    text-align: left;
    background-color: rgb(40,20,20);
}

.dag>h1 {
    background-color: rgb(80,15,15);
    color: orange;
    font-size: 30pt;
}

.tid {
    padding-top: 10px;
    padding-bottom: 20px;
    border-bottom: solid 1px rgb(70,40,20);
    text-align: center;
}

.tid img {
    width: 100%;
    max-width: 400px;
}
```

Her lar vi `<main>`-taggen få flex slik at vi lettere kan lage layout for dagene. For små skjermer legger vi dem ovenfor hverandre ved å angi `wrap` og sette 100 % bredde på *dag*-klassen.

5. For større skjermer vil vi ha oversikten over dagene i kolonner. Da trenger vi heller ikke dag-knappene på toppen. Legg til følgende kode nederst i CSS-fila (*programstil.css*):

```
@media all and (min-width: 1024px) {

    #dagmeny {
        display: none;
    }

    main {
        flex-wrap: nowrap;
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
        justify-content: space-between;
    }

    .dag {
        flex: auto;
        margin-left: 10px;
    }
}
```

Her skjuler vi først knappene. Deretter setter vi `nowrap` slik at dagene kan ligge ved siden av hverandre (i kolonner). Vi setter `justify-content` til `space-between` og `flex` til `auto` slik at dagene fordeler seg med lik bredde. Vi lage også en liten marg for å skille bedre mellom kolonnene:



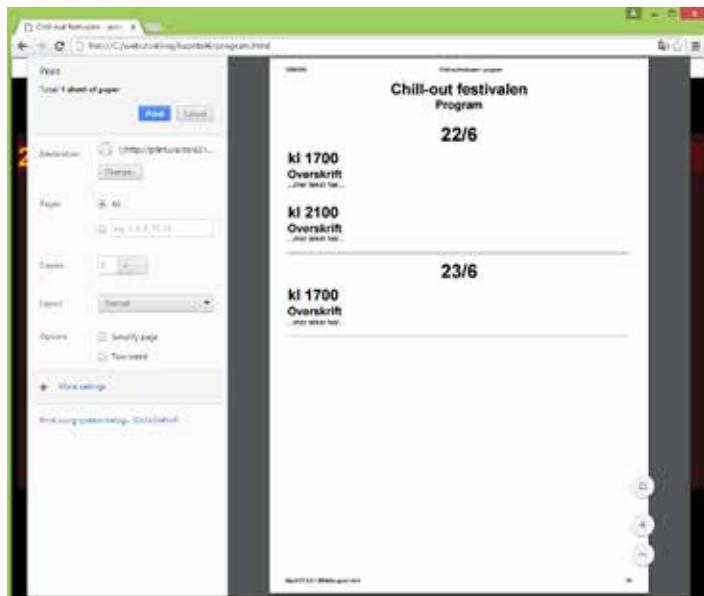
Vær oppmerksom på at hvis festivalen går over mange dager får vi mange kolonner som blir svært smale. Da bør vi overveie å endre layouten slik at dagene får en minimumsbredde. Hvis de ikke får plass i bredden kan de brytes over på neste «linje» (wrap).

6. Til slutt skal vi lage en utskriftsvennlig versjon av programmet. Legg til følgende kode i CSS-fila (*programstil.css*):

```
@media print {  
  
    body {  
        color: black;  
    }  
  
    #menyen, #dagmeny, .tid img {  
        display: none;  
    }  
  
    .dag {  
        text-align: center;  
        border-bottom: solid 1px grey;  
    }  
  
    .dag>h1 {  
        color: black;  
    }  
  
    .tid {  
        text-align: left;  
        border: none;  
    }  
}
```

Her setter vi fargen (tekstfarge) til svart og skjuler både menyer og bilder (hvis du har lagt inn lyd eller video bør du skjule dette også). Vi velger å sentrere dagoverskriften og la resten av teksten være venstre-justert. Vi tar også bort kantlinjene for hvert tidspunkt og lar de i stedet komme for hver dag.

Hvis vi hadde hatt mye innhold bør vi overveie å redusere størrelsen på teksten (**font-size**) slik at vi får plass til mer tekst på sida (brukeren sparer papir):



7 Publisering

I dette kapitlet vil du lære

- om søkemotoroptimalisering
- om måling av nettstedstrafikk
- å annotere informasjon med Microdata
- å integrere sosiale medier
- å markere informasjon for deling i sosiale medier
- om tilgjengelighet
- om RSS

Til nå har vi sett på hvordan du teknisk sett lager en nettside som har innhold og design ved hjelp av HTML og CSS. Dette kapittelet vil ta for seg en del temaer som er verdt å tenke på i forbindelese med at nettsider skal publiseres og bli tilgjengelig for andre.

Selv om vi nå går gjennom disse temaene etter at vi er ferdig med den grunnleggende introduksjonen til HTML og CSS betyr ikke det at disse temaene er noe man gjør helt på tampen av en utviklingsprosess. De fleste av disse temaene bør være med helt fra planleggingstadiet.

Mange av temaene vi presenterer i dette kapittelet har også en side ved seg som ikke er teknisk og med koder. De teoretiske sidene av disse temaene, slik som søkemotoroptimalisering, er beskrevet mer utdypende i f.eks. *IT-1 Basisbok* (3. utgave).

Enkelte temaer i dette kapittelet har potensiale til å endre seg mye etter at boka er trykket. Eventuelle oppdateringer av innholdet vil du finne på www.it1.no.

Søkemotoroptimalisering

Vi skal her kun ta opp enkelte mer tekniske deler av søkemotoroptimalisering. Hva det er, og de mer «teoretiske» sidene av det finner du beskrevet i basisboka.

Nettsiders struktur

Et av de viktigste grepene man kan gjøre med tanke på søkemotoroptimalisering er å lage nettsider med god semantisk struktur. Vi har alt omtalt de semantiske taggene, slik som `<main>`, `<nav>` og `<section>`, samt formateringstagger slik som ``, `<h1>` og ``. Alle disse hjelper søkemotoren med å forstå innholdets betydning.

I tillegg til disse taggene er det bl.a. definert opp en tagg som kalles `<time>`. Denne taggen kan markere et tidspunkt, og så forklare hva dette tidspunktet er i et maskinlesbart format.

Tenk hvor mye vi mennesker kan forstå av utsagn slik som « neste gang » og « bursdagen min ». Dette vil maskiner tolke som ren tekst. Dato og tid skrives også på mange formater, se bare på forvirringen mellom norsk og amerikansk format. Også tidssoner kan skape forvirring. Derfor er det viktig å ha et maskinlesbart format som alle er enige om.

Noen eksempler på bruk av `<time>` er:

```
<p>Det skal vi lære <time datetime="2017-12-14T09:30:15">neste time</time></p>
<p>Det gjorde jeg <time datetime="2017-03-31">31. mars i år</time></p>
```

En annen kjekk tagg er `<abbr>` som forteller hva en forkortelse står for. Ofte går dette klart frem av nettsidens kontekst, men for en søker er det ikke alltid like lett å vite om f.eks. *NRK* er brukt i betydningen *Norsk riksringkasting* eller referer til aksjekoden til den engelske banken *Northern Rock*.

```
<p>Jeg har betalt lisens til <abbr title="Norsk riksringkasting">NRK</abbr></p>
```

Tilleggsinformasjon

I tillegg til struktur i nettsidens informasjon kan vi også inkludere tilleggsinformasjon for søkermotorer i nettsiden. Vanligst er det å inkludere en `robots.txt`-fil og et `XML sitemap`.

`robots.txt`

Ved å plassere en ren tekstfil kalt `robots.txt` i toppkatalogen til domenet, vil søkermotorene oppdage og lese innholdet i denne fila. Altså leter søkermotorene etter `http://www.domenenavn.no/robots.txt`

Vanligst er det å legge informasjon i denne fila som beskriver hvilke deler av



nettsiden vi ikke ønsker at søkemotorene skal indeksere. Fila er forholdsvis enkelt oppbygd, og ser i sin mest vanlige utgave slik ut. Her sier vi at */medlem*, */admin* og */priv/test.html* ikke skal indekseres av noen søkemotorer.

```
User-agent: *
Disallow: /medlem
Disallow: /admin
Disallow: /priv/test.html
```

Husk at denne fila er veiledende. Det er ingen garanti for at søkemotoren følger dine retningslinjer. Husk også at du aldri må legge ut informasjon på en webserver som skal være hemmelig. Fila *robots.txt* skal kun brukes for å skjule ting det vil være unødig, og ikke skadelig, om søkemotorene indekserer. En hacker vil ellers lett kunne lese *robots.txt* for å finne de interessante tingene.

XML sitemap

Et *XML sitemap* fungerer litt som *robots.txt*, men viser i stedet hvilke deler av nettsiden vi ønsker at søkemotorene skal indeksere. Fila heter vanligvis *sitemap.xml*, og plasseres på samme sted som *robots.txt*.

Selv om søkemotorer før eller siden vil indeksere et helt webområde, vil et XML sitemap gjøre prosessen raskere. Det vil også tilby søkemotorene tilleggsinformasjon om oppdateringsfrekvens og viktighet.

Et XML sitemap ser typisk slik ut:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>http://www.domenenavn.no/index.html</loc>
        <lastmod>2016-10-24</lastmod>
        <changefreq>daily</changefreq>
        <priority>1.0</priority>
    </url>
    <url>
        <loc>http://www.domenenavn.no/omoss.html</loc>
        <lastmod>2016-10-13</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.3</priority>
    </url>
</urlset>
```

Som du ser setter vi inn en **<url>**-blokk for hver nettside. Taggen **<lastmod>** forteller når denne nettsiden sist ble oppdatert. Taggen **<changefreq>** forteller hvor ofte vi forventer å oppdatere nettsiden (*always*, *hourly*, *daily*, *weekly*, *monthly*, *yearly* og *never*) og taggen **<priority>** forteller hvor viktig nettsiden er for webområdet i dine øyne (0.0 = uviktig, 1.0 = viktigst).

Du kan få hjelp av verktøyet på <http://www.xml-sitemaps.com> for å raskt lage et utkast til XML-sitemap til din nettside.

TIPS

Analyse

Når vi har satt opp en nettside, og gjort så godt vi kan med søkemotoroptimalisering, er det viktig å ha en analyse av hvordan nettsiden blir besøkt og benyttet, samt hvordan søkermotorene oppfatter nettsiden. Vi skal her ta for oss to ulike verktøy fra Google med navn *Google Analytics* og *Google Webmaster Tools*. For å benytte disse verktøyene må du ha en Google-konto.

Google Webmaster Tools

Verktøyet Google Webmaster Tools hjelper oss til å forstå å påvirke hvordan søkermotoren oppfatter nettstedet vårt.

Verktøyet finner du på nettadressen: <https://www.google.com/webmasters/tools>

For å komme i gang med verktøyet må vi registrere vår nettside. Dette gjør du gjennom knappen **Add a property**. Etter å ha skrevet inn URL-en til nettstedet må du verifisere at det er du som er eieren. Dette kan du gjøre på flere måter, men den enkleste er å laste ned en spesiell fil som Google produserer, og legge i webområdet. Bare du som eier av webområdet skal i teorien ha anledning til å plassere denne fila der.

Verify your ownership of <http://www.help.us>. [Learn more](#).

Recommended method Alternative methods

Recommended: HTML file upload

Upload an HTML file to your site.

- Download this HTML verification file: [google1c3330006c45a8f.html]
- Upload the file to <http://www.help.us/>
- Confirm successful upload by visiting <http://www.help.us/google1c3330006c45a8f.html> in your browser.
- Click Verify below.

To stay verified, don't remove the HTML file, even after verification succeeds.

I'm not a robot

[VERIFY](#) [Wait now](#)

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Etter at nettsiden er verifisert kan du åpne egenskapene for nettsiden fra forsiden av Webmaster Tools. Vær klar over at det tar litt tid før informasjonen begynner og vises.

The screenshot shows the Google Search Console dashboard. On the left, there's a sidebar with links like 'Dashboard', 'Messages (0)', 'Search Appearance', 'Search Traffic', 'Google Index', 'Crawl', 'Security Issues', and 'Other Resources'. The main area has a section titled 'New and important' with a message: 'Improve the search presence of http://www.datalikhetshetsboka.no/'. Below that is the 'Current Status' section, which includes 'Crawl Errors' (with 0 errors), 'Search Analytics' (showing 921 Total Clicks), 'Sitemaps' (with 80 URLs submitted and 78 indexed), and 'URLs, Errors' (with 5 Not found). At the bottom, there's a note: '© 2010 Google Inc. - Webmaster Central - Terms of Service - Privacy Policy - Search Console Help'.

I dette verktøyet vil du bl.a. få tilgang til å se hvor mange som får opp din nettside i søker treffene, og hva de søker etter. Dette er en god indikasjon på om man har funnet riktige nøkkelord til bruk i nettsiden.

Search Analytics

Analyze your performance on Google Search. Filter and compare your results to better understand your user's search patterns. [Learn more](#)

This part of the screenshot shows the 'Search Analytics' section. It features a filter bar with checkboxes for 'Clicks' (checked), 'Impressions' (checked), 'CTR' (unchecked), and 'Position' (checked). Below the filter bar are several dropdowns for filtering by 'Queries', 'Pages', 'Countries', 'Devices', 'Search Type', and 'Dates'. Under these filters, there are summary statistics: 'Total clicks: 921', 'Total impressions: 21,418', and 'Avg. position: 10.1'. A line chart below these stats tracks the performance over time for three different queries, represented by red, green, and blue lines. At the bottom, there's a table showing the top search queries with their Clicks, Impressions, and Position:

Queries	Clicks	Impressions	Position
1 pointworld svindel ↗	36	248	5.2 ↗
2 quizconsul svindel ↗	23	57	1.3 ↗
3 falske facebook profiler ↗	17	74	2.9 ↗
4 pointworld ↗	13	355	9.2 ↗

Vi kan også se hvem som linker til oss og hvilke nettsider søkermotoren faktisk har indeksert. Vi kan til og med se hvilke nøkkelord som søkermotoren synes er mest fremtredende på nettsiden. I tillegg forteller verktøyet oss om feil og mangler Google har funnet i nettsiden.

Vi har ikke anledning til å gå gjennom hele Webmasters tools i denne boka, men her er det mye interessant statistikk og mange innstillinger vi kan sette.

Bing har tilsvarende verktøy på adressen: <http://www.bing.com/toolbox/webmaster>

Google Analytics

Google Analytics lar deg få en oversikt over å analysere alle besøk til din nettside.

Verktøyet finner du på adressen <http://analytics.google.com>.

For å komme i gang må du registrere nettsiden på omtrent samme måte som for Google Webmaster Tools. Du velger **Create New Account** under **Admin**-menyen.



I stedet for å få en fil du skal laste ned på weområdet får du derimot en liten snutt med JavaScript-kode som skal inkluderes i alle nettsider i webområdet som du ønsker å analysere. Det er denne JavaScript-koden som gjør at Google får tilgang til informasjon om nettsidens besøk.

Website tracking

This is the Universal Analytics tracking code for this property.

To get all the benefits of Universal Analytics for this property, copy and paste this code into every webpage you want to track.

```
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments),i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/analytics.js','ga');

ga('create', 'REDACTED', 'auto');
ga('send', 'pageview');

</script>
```

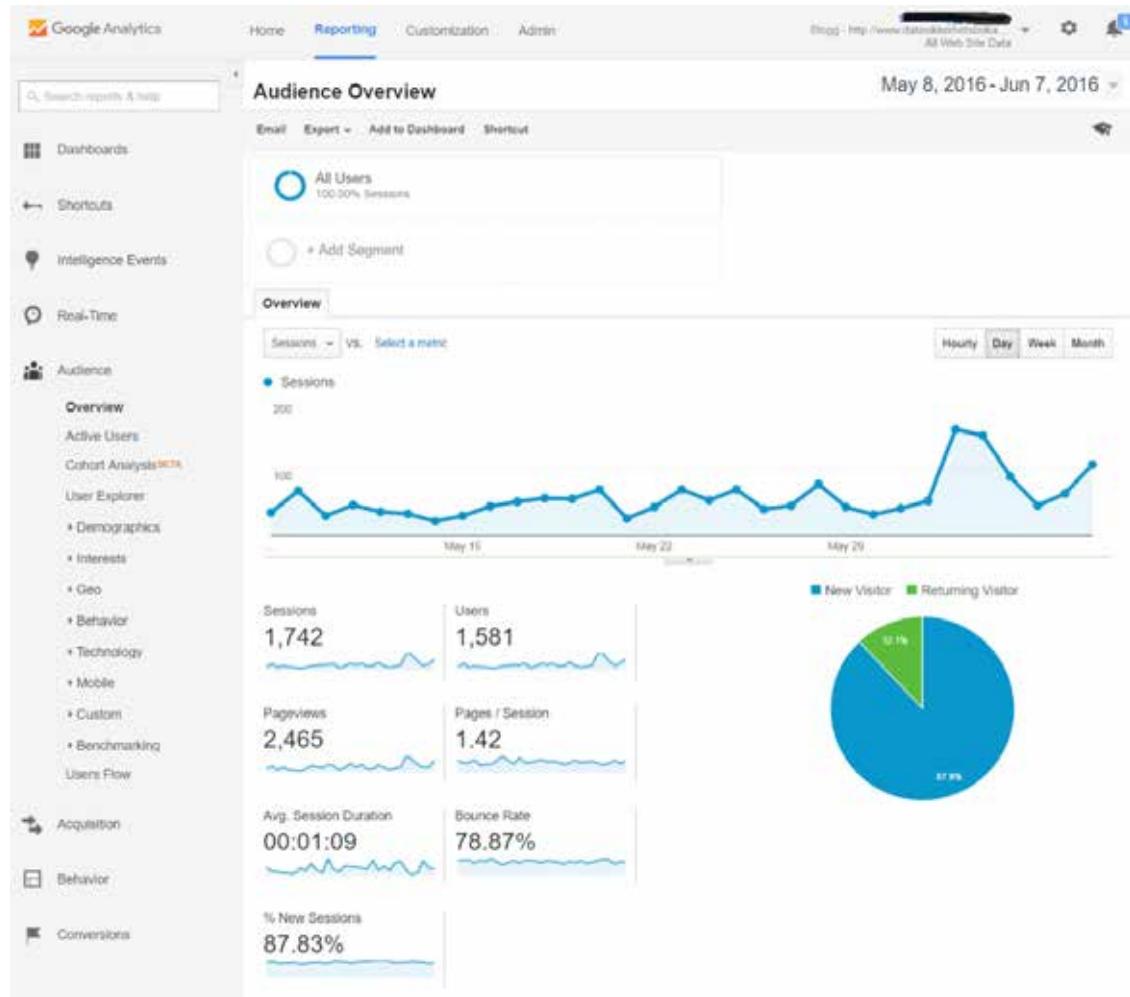
TIPS

Google har også lansert en demo-konto du kan leke deg med, dersom du ikke har en egen nettside med mye trafikk:

<https://analytics.google.com/analytics/web/demoAccount>

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Etter at nettsiden er satt opp kan vi benytte Google sitt avanserte analyseverktøy for å hente ut informasjon om de besøkende. Merk deg at det kun er besøk etter at nettsiden er registrert som blir fanget opp.



Vi kan få statistikk over alt fra hvor lenge brukere var på nettsiden til hvilke kjønn brukerne har og hvor de bor. Vi kan under seksjonen **Acquisition** også finne ut hvor brukerne kom fra. Altså hvilke søkeord, sosiale medier eller refererende nettsider det var som sendte de til oss.

Svært mye av statistikken om såkalt demografi kan Google hente ut fordi besøkende på din nettsiden samtidig er innlogget i noen av Googles andre tjenester.

TIPS



All denne statistikken kan vi også sette sammen i rapporter av flere dimensjoner. Vi kan dermed finne ut om jenter tilbringer lengre tid på nettsiden enn gutter eller om type enhet de besøkende benytter varierer med alderen. I mer avanserte nettsider som har «butikker» kan vi også koble inn salg. Dermed kan vi se hvilke brukere som står for mest salg, og hvilken oppførsel disse har.

Google Analytics er altså et svært avansert verktøy, men allikevel forholdsvis enkelt i bruk. Her er det bare å leke seg blant alle de data som finnes, og forsøke finne ut hvem din nettsides besøkende er.

Sunn skepsis

I vår omtale av Google Webmasters Tools og Google Analytics er vi svært entusiastiske over mulighetene disse verktøyene gir. Vi må imidlertid også advare om at ved å benytte disse verktøyene gir vi Google svært mye ekstra informasjon om våre nettsider og våre besøkende.

Ønsker vi egentlig at Google skal vite hvor mange som besøker nettsidene våre, og hvem de er? Og for større nettsider, vil vi egentlig at Google skal vite hvor mye salg vi har gjennom nettsiden?

Vi lar dette sprøsmålet stå litt åpent, og opp til deg å vurdere.

Microdata

Det nye formatet Microdata lar deg markere innholdstype og angi hvilken betydning ulike deler av innholdet har. Microdata blir bruk til maskinell lesing av informasjon, og er bl.a. stadig viktigere for søkemotorer. Vi skal her først og fremst fokusere på søkemotorers bruk av Microdata.

Microdata er en del av HTML5-standarden og består av noen ekstra attributter alle HTML5-tagger kan få. Disse attributtene er `itemscope`, `itemtype`, `itemprop`, `itemid` og `itemref`. Vi skal kun benytte de tre første i denne boka, da bruken av `itemid` og `itemref` er noe mer avansert.

Tenk deg at du har følgende tekst på nettsiden, som beskriver et event (hendelse):

```
<main>
<h1>Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017.
Dugnaden starter 18.00 og holder på til 22.00.</p>
<p>Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

For at en maskin nå skal forstå at 18.00 er starttidspunkt og at 3452 er postnummeret til stedet dette foregår, må vi *annotere* (markere) denne informasjonen.

Det første vi må gjøre er å legge inn attributtet **itemscope** i en tagg som omslutter informasjonen. Attributtet **itemscope** forteller at alt som kommer inne i denne taggen handler om samme type informasjon. Her har vi alt en **<main>** tagg som passer til dette bruket. Ellers kunne vi satt in **<div>** eller ****.

```
<main itemscope="itemscope">
<h1>Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017.
Dugnaden starter 18.00 og holder på til 22.00.</p>
<p>Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

Deretter må vi finne et *vokabular* som passer den typen informasjon vi skal markere. Det finnes en mengde ulike vokabular, men de vi finner beskrevet på *schema.org* er utarbeidet og støttet av alle de store søkermotorene. På *schema.org* finner vi at vokabularet **Event** kan passe:

The screenshot shows the schema.org homepage with a red header. Below it, a navigation bar with links to Home, Schemas, and Documentation. The main content area has a title "Event" and a breadcrumb trail "Thing > Event". A descriptive text follows, mentioning that an event is a happening at a certain time and location, such as a concert, lecture, or festival. It notes that ticketing information may be added via the 'offers' property. Repeated events may be structured as separate Event objects. Below this is a usage statistic: "Usage: Between 100,000 and 250,000 domains" and a "[more...]" link. The central part of the page is a table titled "Properties from Event" with columns for "Property", "Expected Type", and "Description". The table lists various properties with their types and descriptions, such as "actor" (Person), "aggregateRating" (AggregateRating), "attendee" (Organization or Person), "composer" (Organization or Person), "contributor" (Organization or Person), "director" (Person), "duration" (Duration), "endDate" (Date), "eventStatus" (EventStatusType), "inLanguage" (Language or Text), and "location" (Place or PostalAddress).

Property	Expected Type	Description
Properties from Event		
actor	Person	An actor, e.g. in tv, radio, movie, video games etc., or in an event. Actors can be associated with individual items or with a series, episode, clip. Supersedes actors .
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
attendee	Organization or Person	A person or organization attending the event. Supersedes attendees .
composer	Organization or Person	The person or organization who wrote a composition, or who is the composer of a work performed at some event.
contributor	Organization or Person	A secondary contributor to the CreativeWork or Event.
director	Person	A director of e.g. tv, radio, movie, video gaming etc. content, or of an event. Directors can be associated with individual items or with a series, episode, clip. Supersedes directors .
duration	DateTime	The time admission will commence.
duration	Duration	The duration of the item (movie, audio recording, event, etc.) in ISO 8601 date format.
endDate	Date	The end date and time of the item (in ISO 8601 date format).
eventStatus	EventStatusType	An eventStatus of an event represents its status, particularly useful when an event is cancelled or rescheduled.
inLanguage	Language or Text	The language of the content or performance or used in an action. Please use one of the language codes from the IETF BCP 47 standard. See also availableLanguage . Supersedes language .
location	Place or PostalAddress	The location of for example where the event is happening, an organization is located, or where an action takes place.

Det pussige nå er at det ikke er innholdet på denne nettsiden som er det viktigste, men selve URL-en <http://schema.org/Event>. Denne URL-en er det som unikt identifiserer hvilket vokabular vi benytter, og settes som **itemtype** sammen med **itemscope**.

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1>Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017.
Dugnaden starter 18.00 og holder på til 22.00.</p>
<p>Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

Nå er vi klare til å markere de ulike bitene informasjon ved hjelp av **itemprop**-attributtet. Hvilke egenskaper som finnes for det valgte vokabularet er beskrevet på nettsiden til vokabularet. Som vi ser er **Event** en spesifisering av **Thing**, så en del av egenskapene vi ønsker benytte finnes beskrevet der, slik som egenskapen **name**.

name	Text	The name of the item.
------	------	-----------------------

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1 itemprop="name">Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017.
Dugnaden starter 18.00 og holder på til 22.00.</p>
<p>Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

For å markere tidspunktene for start og slutt må vi sette inn en ekstra tagg rundt selve verdiene. Vanligvis benyttes **** til å sette inn markeringstagger, men akuratt for tidspunkt er **<time>** et bedre valg.

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1 itemprop="name">Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-
2017. Dugnaden starter <time datetime="2017-05-04T18:00"
itemprop="startDate">18.00</time> og holder på til <time
datetime="2017-05-04T22:00" itemprop="endDate">22.00</time>.</p>
<p>Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

Selve adressen er litt spesiell ettersom den både er en **location**-egenskap til et **Event**, men også er en egen blokk med informasjon i seg selv gjennom å være en **PostalAddress**. Derfor må vi si at den omsluttende taggen er en ny **itemscope**, samtidig som den er en **itemprop**.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1 itemprop="name">Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017. Dugnaden starter <time datetime="2017-05-04T18:00" itemprop="startDate">18.00</time> og holder på til <time datetime="2017-05-04T22:00" itemprop="endDate">22.00</time>. </p>
<p itemprop="location" itemscope="itemscope" itemtype="http://schema.org/PostalAddress">Adressen er Gateveien 2, 3452 Lilleby</p>
</main>
```

Ettersom vi har hele adressen som én eneste paragraf, må vi bryte opp delene ved hjelp av ****-tagger:

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1 itemprop="name">Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017. Dugnaden starter <time datetime="2017-05-04T18:00" itemprop="startDate">18.00</time> og holder på til <time datetime="2017-05-04T22:00" itemprop="endDate">22.00</time>. </p>
<p itemprop="location" itemscope="itemscope" itemtype="http://schema.org/PostalAddress">
Adressen er <span itemprop="streetAddress">Gateveien 2</span>,
<span itemprop="postalCode">3452</span>
<span itemprop="addressLocality">Lilleby</span>
</p>
</main>
```

Som du ser øker Microdata mengden kode vesentlig. Det er allikevel mange gode grunner til å benytte dette. En av de viktigste pr. i dag er at søkerne plutselig forstår informasjonen, og i ganske nær fremtid kan tillate søk slik som «*Hva skjer i nærheten av meg neste torsdag*» eller «*Ostekake som det tar mindre enn 30 minutter å lage og som har under 400 kcal*». Også rich snippets som vi beskriver straks burde være grunn god nok.



Rich snippets

En annen stor fordel med Microdata er at søkermotorer kan presentere informasjonen i nettsiden på en mer informativ og en mer tiltalende måte i søker treffene. Dette kalles av Google for *rich snippets*.

Best pris på Samsung Galaxy S7 SM-G930F 32GB - Sammenlign ...

[www.prisjakt.no](#) › Telefon & GPS › Mobiltelefoner ▾

★★★★★ Vurdering: 8/10 - 28 stemmer - Fra kr 6 290,00 til kr 8 829,00

30+ elementer - **Samsung Galaxy S7 SM-G930F 32GB** - Viser priser.

Butikk	Merverdi	Inkl. frakt
Proshop.no	Avdrag, faktura, kortbetaling, Rask ...	6 290
PS	Stort utvalg. Lave priser. Norges eldste ...	6 329

Ostekake med gelélokk - Aperitif.no



[www.aperitif.no/oppskrifter/oppskrift/ostekake-med-gelelokk/283312](#) ▾

★★★★★ Vurdering: 5 - 74 stemmer - 4 t 30 min

Til denne **ostekaken** trenger du en rund form, ca 24 cm i diameter. Bunn: Knus kjeksen, gjerne i en food processor. Tilsett smeltet smør og kjør til du får en jevn ...

Andrea Bocelli Tour Dates and Concert Tickets | Eventful

[concerts.eventful.com/Andrea-Bocelli](#) ▾ [Oversett denne siden](#)

Andrea Bocelli tour dates and concert tickets in 2016 on Eventful. Get alerts when Andrea Bocelli comes to your city or bring Andrea Bocelli to your city usi...

søn. 26. jun.	Andrea Bocelli Concert	Stadion Legii, Warsaw, POL
fre. 29. jul.	Andrea Bocelli - Le Cirque ...	Teatro Del Silenzio, Laiatico, ITA
lør. 30. jul.	Andrea Bocelli Lajatico	Teatro Del Silenzio, Laiatico, ITA

Google har laget en liste over hvilke informasjonstyper de støtter for rich snippets på følgende nettside:

<https://developers.google.com/search/docs/guides/mark-up-content>

I tillegg til å markere ren informasjon finnes det også en del Microdata-vokabularer som mer teknisk gjør noe med presentasjonen. En av disse er

<http://schema.org/BreadcrumbList> som kan benyttes for å lage såkalte *breadcrumbs* i søker treffene på større nettsider:

Samsung Galaxy S7 review - CNET

[www.cnet.com](#) › Mobile › Phones ▾ [Oversett denne siden](#)

★★★★★ Vurdering: 4,5 - Vurdering fra Jessica Dolcourt

8. mar. 2016 - So I tested the **Samsung Galaxy S7** in London and Berlin, while colleagues also took it for a spin in San Francisco and Sydney. And you know ...

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

```
<ol itemscope itemtype="http://schema.org/BreadcrumbList">
  <li itemprop="itemListElement" itemscope="itemscope" itemtype="http://schema.org/ListItem">
    <a href="/" itemprop="item">
      <span itemprop="name">Hjem</span>
    </a>
  </li>
  <li itemprop="itemListElement" itemscope="itemscope" itemtype="http://schema.org/ListItem">
    <a href="/mobiltelefoner" itemprop="item">
      <span itemprop="name">Mobiltelefoner</span>
    </a>
  </li>
  <li itemprop="itemListElement" itemscope="itemscope" itemtype="http://schema.org/ListItem">
    <a href="/mobiltelefoner/Samsung" itemprop="item">
      <span itemprop="name">Samsung</span>
    </a>
  </li>
</ol>
```

Verktøy for Microdata

Som du har sett gir Microdata ofte svært omfattende og avansert kode. Det er fort gjort å introdusere feil når vi skriver slik kode. Rent strukturelle feil i koden (manglende slutttagger osv.) fanger den ordinære validatoren til W3C opp. Feil i bruk av vokabularers egenskaper sjekkes derimot ikke av dette verktøyet.

Google har imidlertid laget sitt eget valideringsverktøy som er svært kjekt. Du finner dette verktøyet på adressen:

<https://search.google.com/structured-data/testing-tool>.

Her kan du legge inn kildekode eller en URL, og får så se hvordan Google tolker de strukturerte dataene i nettsiden:



Test your structured data

FETCH URL CODE SNIPPET

```

1 <main itemscope="itemscope" itemtype="http://schema.org/Event">
2   <h1 itemprop="name">Dugnad på fotballbanen</h1>
3   <p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017. Dugnaden
4     starter <time datetime="2017-05-04T18:00">
5       itemprop="startDate">18.00</time> og holder på til <time datetime="2017-
6       05-04T22:00"> itemprop="endDate">22.00</time>. </p>
7   <p itemprop="location" itemscope="itemscope"
8     itemtype="http://schema.org/PostalAddress">
9     Adressen er <span itemprop="streetAddress">Gateveien 2</span>, <span
10    itemprop="postalCode">3452</span> <span
11    itemprop="addressLocality">Lilleby</span>
12  </p>
13 </main>
```

RUN TEST

Explore the [Search Gallery](#). Learn more about this tool.

Google Structured Data Testing Tool

Event

Event	2 ERRORS 6 WARNINGS
name	Dugnad på fotballbanen
startDateTime	2017-05-04T18:00:00
endDateTime	2017-05-04T22:00:00
location	PostalAddress
streetAddress	Gateveien 2
postalCode	3452
addressLocality	Lilleby
name	A value for the name field is required.
location	Field location may not be empty.
description	The description field is recommended. Please provide a value if available.
image	The image field is recommended. Please provide a value if available.
offers	The offers field is recommended. Please provide a value if available.
performer	The performer field is recommended. Please provide a value if available.
url	The url field is recommended. Please provide a valid url.

Som du ser produserer vår kode feil, da Google sin sjekk er veldig streng. Det den etterlyser er en **name**-egenskap også for adressen, noe vi ikke har. Vi kan velge

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

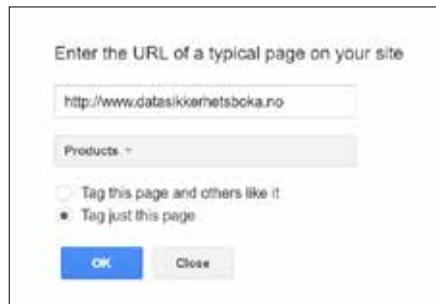
å se bort i fra feilen, eller legge inn data for dette. Enten som faktiske data, eller en tom tagg:

```
<main itemscope="itemscope" itemtype="http://schema.org/Event">
<h1 itemprop="name">Dugnad på fotballbanen</h1>
<p>Du inviteres herved til dugnad på fotballbanen den 4/5-2017. Dugnaden starter <time datetime="2017-05-04T18:00" itemprop="startDate">18.00</time> og holder på til <time datetime="2017-05-04T22:00" itemprop="endDate">22.00</time>.</p>
<p itemprop="location" itemscope="itemscope" itemtype="http://schema.org/PostalAddress">
Adressen til <span itemprop="name">fotballbanen</span>
er <span itemprop="streetAddress">Gateveien 2</span>, <span itemprop="postalCode">3452</span> <span itemprop="addressLocality">Lilleby</span>
</p>
</main>
```

Et annet kjekt verktøy fra Google er et hjelpeverktøy der vi kan angi en nettside uten markering, og få hjelp til å markere denne. Verktøyet finner du som en del av Google Webmaster Tools, som vi har sett på tidligere:

<https://www.google.com/webmasters/tools/data-highlighter>

Først angir man nettsiden man vil markere:



Deretter kan vi markere elementer (slik vi markerer tekst ellers), høyreklikke og velge hvilken egenskap informasjonen skal knyttes til:

Til slutt kan vi få Google til å vise HTML-koden med Microdata-tagging, slik at vi kan benytte denne videre som vår egen kode.

Kobling mot sosiale medier

Sosiale medier blir en stadig viktigere kommunikasjonskanal, og det er viktig å koble våre nettsider mot disse. Vi skal her ta for oss to ulike metoder for slik kobling; Det å optimalisere nettsidene for deling i sosiale medier, og det å inkludere informasjon fra sosiale medier i nettsidene.

Optimalisere for deling

Når vi deler en URL i sosiale medier, slik som Twitter, Facebook og Pinterest, vil

disse tjenestene forsøke å hente ut informasjon fra nettsidene. Denne informasjonen benyttes dels til å lage informative forhåndsvisninger av innholdet, og dels for å kategorisere og prioritere innholdet for visning i andre brukeres nyhetsstrømmer. Det er derfor svært viktig at vi optimaliserer nettsiden slik at den kan tolkes på best mulig måte.



De ulike tjenestene har ulike krav til hvordan nettsiden skal formateres, så her må vi gjøre flere ting.

Facebook

Facebook jobber med informasjon i formatet *Open Graph*. Dette går ut på at vi legger inn en rekke `<meta>`-tagger i `<head>`-seksjonen av nettsiden:

Her finnes det en del «standardelementer» som starter med `og`: og fortsetter med elementnavnet.

```
<meta property="og:title" content="Datasikkerhet - Ikke bli
svindlerens neste offer | Ny bok" />
<meta property="og:description" content="En ny bok om datasikkerhet
for folk flest. Boka hjelper deg til ikke å bli et offer for hackere og
svindlere ved hjelp av enkle forklaringer og dagligdagse eksempler." />
<meta property="og:type" content="book" />
<meta property="og:locale" content="nb_NO" />
<meta property="og:url" content="http://www.datasikkerhetsboka.no" />
<meta property="og:image" content="http://www.datasikkerhetsboka.no/
datasikkerhet.jpg" />
```

Det finnes også en del spesifikke tagger til det temaet nettsiden handler om, slik som **book:** i eksemplet vi her viser:

```
<meta property="book:author" content="tomheine"/>
<meta property="book:author" content="christian.heide.737"/>
<meta property="book:isbn" content="978-82-05-48026-1"/>
<meta property="book:release_date" content="2015-11-20"/>
<meta property="book:tag" content="datasikkerhet"/>
<meta property="book:tag" content="datakriminalitet"/>
<meta property="book:tag" content="hacker"/>
<meta property="book:tag" content="hackere"/>
<meta property="book:tag" content="hacking"/>
<meta property="book:tag" content="svindler"/>
<meta property="book:tag" content="svindlere"/>
<meta property="book:tag" content="bok"/>
```

Du finner en oversikt over hele Open Graph-standarden på følgende nettside:

<http://ogp.me/>

Facebook har også utviklet et verktøy for å teste og validere dine Open Graph-koder: <https://developers.facebook.com/tools/debug>

Twitter

Twitter baserer seg på et lignende system som Facebook, men i stedet for Open Graph benytter de sitt eget *Twitter Cards*:

```
<meta name="twitter:card" content="summary_large_image" />
<meta name="twitter:site" content="@tomhnatt" />
<meta name="twitter:creator" content="@tomhnatt" />
<meta name="twitter:title" content="Datasikkerhet - Ikke bli
svindlerens neste offer | Ny bok" />
<meta name="twitter:description" content="En ny bok om datasikkerhet
for folk flest. Boka hjelper deg til ikke å bli et offer for hackere og
svindlere ved hjelp av enkle forklaringer og dagligdagse eksempler." />
<meta name="twitter:image" content="http://www.datasikkerhetsboka.no/
datasikkerhet.jpg" />
```

Du finner en oversikt over hele Twitter Cards-standarden her:

<https://dev.twitter.com/cards/overview>

Twitter har også laget sitt eget valideringsverktøy:

<https://cards-dev.twitter.com/validator>



Google+

Google+ har valgt en annen tilnærming til markeringen av informasjon, og benytter seg ganske enkelt av Microdata (selv om også Open Graph kan benyttes). Dermed er det nok å ha markert informasjonen i selve nettsiden med Microdata for at Google+ skal hente ut denne. Her er det i hovedsak egenskapene i vokabularet <http://schema.org/Thing> som benyttes. Alle andre vokabularer er en utvidelse av dette.

De viktigste egenskapene å ha med er **name**, **image**, **description** og **url**.

Du kan lese mer om markering for Google+ her:

<https://developers.google.com/+/web/snippet/>

Pinterest

Pinterest leser data angitt med både Microdata og Open Graph. Du kan lese mer om såkalte *Rich Pins* her: <http://business.pinterest.com/rich-pins/>

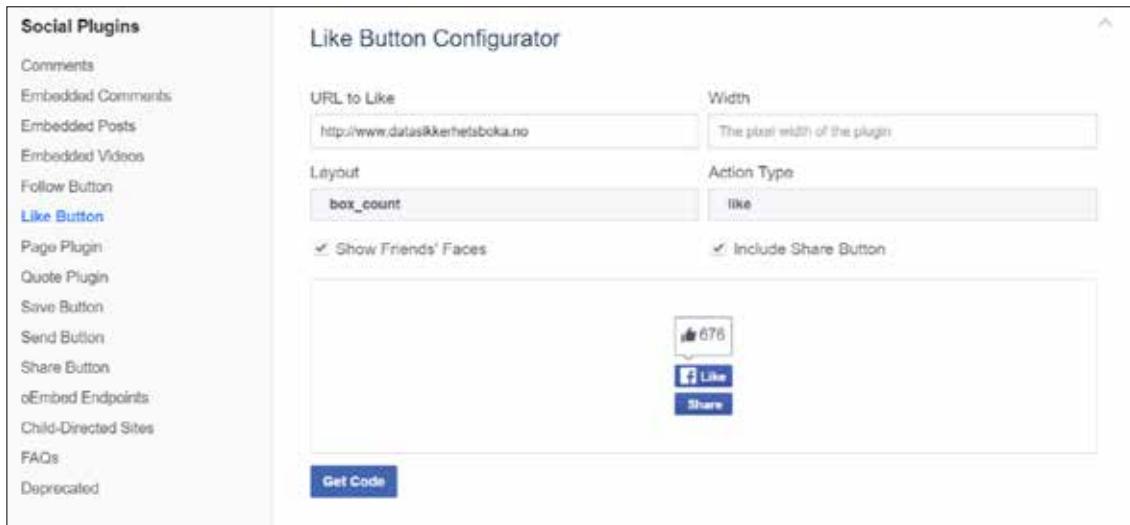
Inkludere data fra sosiale medier

Som tidligere nevnt kan vi også gå motsatt vei, og inkludere informasjon fra sosiale medier i nettsiden. Dette krever i utgangspunktet noe programmering eller mer avansert HTML, men heldigvis har tjenestene laget verktøy som produserer denne koden for oss på en svært enkelt måte.

Facebook

Facebook har et stort bibliotek med ulike såkalte *social plugins* vi kan benytte i vår nettside. Du finner verktøy for å lage koden på
<https://developers.facebook.com/docs/plugins>

Her er det bare å velge type plugin i menyen, og så fylle ut feltene:



The screenshot shows the 'Like Button Configurator' tool on the Facebook Developers website. On the left, there's a sidebar with a list of social plugins: Social Plugins, Comments, Embedded Comments, Embedded Posts, Embedded Videos, Follow Button, Like Button (which is selected), Page Plugin, Quote Plugin, Save Button, Send Button, Share Button, oEmbed Endpoints, Child-Directed Sites, FAQs, and Deprecated. The main area is titled 'Like Button Configurator'. It has fields for 'URL to Like' (set to 'http://www.datasikkerhetsboka.no'), 'Width' (set to 'The pixel width of the plugin'), 'Layout' (set to 'box_count'), 'Action Type' (set to 'like'), and checkboxes for 'Show Friends' Faces' and 'Include Share Button'. Below these is a preview window showing a like button with a count of 676 and 'Like' and 'Share' buttons. At the bottom is a 'Get Code' button.

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Deretter trykker man **Get Code**, og all koden genereres sammen med forklaringer på hvor de skal plasseres:

Step 1: Choose your App ID and Language

Datasikkerhetsboka.no English (UK)

Step 2: Include the JavaScript SDK on your page once, ideally right after the opening <body> tag.

```
<div id="fb-root"></div>
<script>(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s) [0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src =
  "//connect.facebook.net/en_GB/sdk.js#xfbml=1&version=v2.6&appId=154078310624351
  8";
  fjs.parentNode.insertBefore(js, fjs);
  (document, 'script', 'facebook-jssdk'));</script>
```

Step 3: Place this code wherever you want the plugin to appear on your page.

```
<div class="fb-like" data-href="http://www.datasikkerhetsboka.no" data-
layout="box_count" data-action="like" data-show-faces="true" data-share="true">
</div>
```

Twitter

Twitter har et tilsvarende system for å lage såkalte *widgets/skjemaelementer* av ulike typer på nettadressen <https://twitter.com/settings/widgets>

Skjermelementer
Opprett og administrér skjermelementer.

Du har ingen skjermelementer.

Opprett nytt

- Profil
- Liker
- Liste
- Samling
- Seik

Etter å ha valgt type element, guider veiviseren deg gjennom resten av prosessen.

Tilgjengelighet

Når vi snakker om tilgjengelighet og nettsider tenker vi stort sett på svaksynte, men tilgjengelighet er så utrolig mye mer. Tilgjengelighet gjelder vel så mye ulike typer enheter (f.eks. mobiler), språk, båndbredde (hastighet på nettforbindelse) og versjoner av programvare.

Det pussige er at det meste som er viktig å tenke når det gjelder tilgjengelighet har vi alt lært tidligere i denne boka. Tilgjengelighet går nemlig i stor grad ut på at maskiner skal klare å forstå innholdet og presentere det på en måte som passer brukeren.

Ved å sørge for at nettsiden har en god struktur, at vi benytter semantiske tagger, at vi skiller innhold og design samt at vi har nettsider som følger prinsippet om responsive web design, har vi faktisk det meste på plass.

Enkle ting slik som `<abbr>`-taggen gjør f.eks. at nettslere alt i dag kan vise en hjelpetekst når vi holder musepekken over ordet for de som ikke forstår fagterminer, og gjør også så at bl.a. Google translate oversetter riktig.

Vi betaler lisens til NRI [Norsk riksringkasting](#) apparat.

Vi skal imidlertid ta for oss et par ekstra momenter når det gjelder skermlesere og syntetisk tale.

Syntetisk tale

En *skjermleser* er en spesiell nettleser som leser opp informasjonen og hjelpetekst for hva informasjonen er (slik som en link) ved hjelp av *syntetisk tale*. De kan også hjelpe til med navigasjon i nettsiden.

For at en syntetisk tale skal fungerer den helt avhengig av at vi har formatert nettsiden korrekt med tagger slik som `<h1>`, `<abbr>`, `` osv. De baserer seg også i stor grad på de semantiske taggene for innholdsoppdeling og navigasjon, slik som `<main>` og `<nav>`.

I tillegg til dette finnes det en utvidelse til CSS som kalles *CSS speech*. Vi kan inkludere et eget stilark som forteller hvordan nettsiden skal «høres», på samme måte som vi ellers inkluderer et stilark for hvordan nettsiden skal sees:

```
<link rel="stylesheet" type="text/css" href="tale.css" media="speech" />
```

I dette stilarket kan vi så fortelle noe om stemmetyper og volum, der vi ellers ville fortalt om fonttyper og skriftstørrelse:

```
h1 {  
  voice-family: paul;  
  cue-before: url('ding.mp3');  
  voice-volume: medium 6dB;  
}  
  
p {  
  voice-family: female;  
  voice-pitch: high;  
  voice-volume: -3dB;  
}
```

Dessverre er slike skjermlesere i dag ofte dyre og ikke spesielt gode. Dette vil nok snart endre seg etter hvert som det å få lest opp en nettside blir vanlig å gjøre mens man kjører bil eller trener, på samme måte som lydbøker er blitt svært mye vanlige i dag enn for et par år siden.

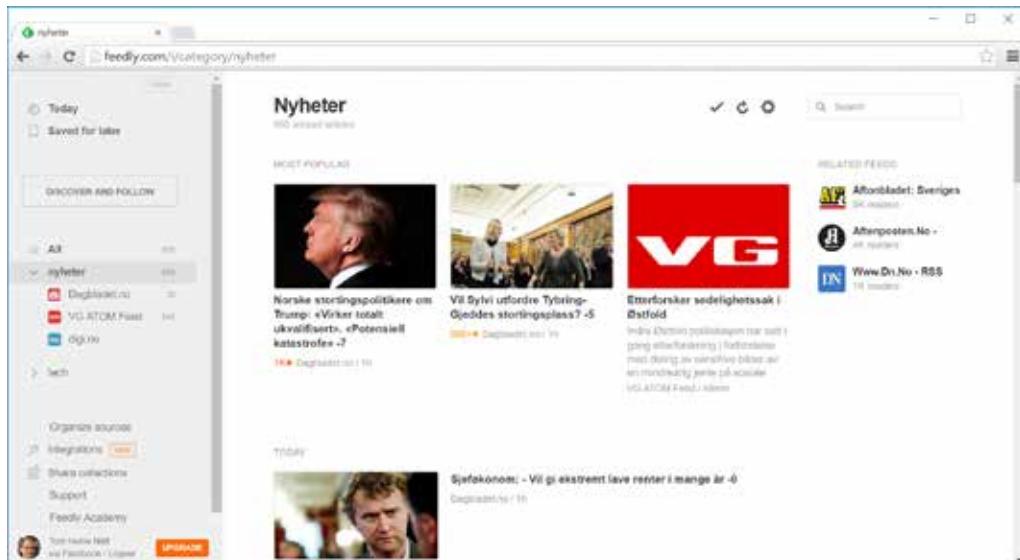
Du kan lese mer om CSS Speech-standarden på <https://www.w3.org/TR/css3-speech/>



Annet

RSS

En forholdsvis ukjent teknikk for mange er RSS (*Really Simple Syndication*). Dersom en nettside støtter RSS kan vi holde oss oppdatert på nyheter fra denne og andre nettsider i en felles RSS-leser. Et eksempel på en slik leser er *Feedly* (www.feedly.com). Stort sett alle nettavisar og blogger har en RSS-strøm.



Vi kan forholdsvis enkelt lage vår egen RSS-feed til vårt webområde, slik at den kan benyttes i alle RSS-lesere.

Først må vi lage selve RSS-dokumentet. Dette består av noen standardiserte felter, og så en rekke blokker av typen <item>, der én blokk er én nyhetssak.

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Agurkavisa</title>
    <link>http://www.agurkavisa.no</link>
    <description>En avis med agurknyheter...</description>
    <item>
      <title>Mann stjal agurk</title>
      <link>http://www.agurkavisa.no/viewpage.php?id=56</link>
      <description>En mann stjal en agurk i en butikk...</description>
      <pubDate>Mon, 17 Aug 2016 16:23:21 GMT</pubDate>
    </item>
    <item>
      <title>Merkelig agurk funnet i butikk</title>
      <link>http://www.agurkavisa.no/viewpage.php?id=72</link>
      <description>En agurk med to ender funnet i en Rema-butikk i Oslo</description>
      <pubDate>Tue, 18 Aug 2016 10:13:21 GMT</pubDate>
    </item>
  </channel>
</rss>
```

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Du kan lese mer om RSS-standarden og de ulike elementene den er bygd opp av her: <https://validator.w3.org/feed/docs/rss2.html>

Etter at vi har lagret RSS-dokumentet som f.eks. *rss.xml*, må vi indikere at vi har et RSS-dokument i nettsiden vår. Dette gjøres gjennom følgende tagg i **<head>**-seksjonen av nettsiden:

```
<link href="rss.xml" type="application/rss+xml" rel="alternate" title="Min RSS-feed" />
```

Vi kan også installere en nettleserutvidelse som viser oss at nettsiden vi besøker har RSS ved hjelp av et lite ikon på verktøylinjen til nettleseren. Chrome har sin «Utvidelse for RSS-abonnement».



Denne utvidelsen tillater også at vi leser og kan abonnere på RSS rett i nettleseren:

Innmatting for VG RSS
Abonnement på denne innmattingen ved bruk av: Abonnér nå
 Bruk alltid denne leseren for abonnement på innmattinger.

VW antyder at de vil slutte med dieselmotorer
Vestsvenskens største bilprodusent Martinus Knutler antyder at det kan bli hent utslip av dieselmotorer etter avslutningen av utslippsuka i fjor høst.

Elisabeth Andreassens mann bisesettes
Tor Andreassens bisesset fra Ulvene Kroka i 14. En av de første som kom var Sven Nordahl og kona Torill.

Lewandowski er veltrent som få takket være kona
Blant tyske lagkamerater i Borussia Dortmund og na Bayern München blir Robert Lewandowski (27) bare kalt for «The Body». Det kan den polske lagkaptenen natt annet takke sin kone for.

Etterforsker sedelighetssak i Østfold
Innre Østfold politiet har satt i gang etterforskning i forbindelse med døring av sensitiivt blod av en mindreårig jente på sosiale medier.

Holst Enger klar for Tour de France
Norge først i lønnet årets Tour de France. I dag får også Sondre Holst Enger (22) grønt lys av laget sitt.

Hodgson slaktes: Et av de mest bizarre trekkene en England-manager noensinne har gjort
England manager Roy Hodgson (58) får massiv kritikk i britiske medier for valgene han gjorde før kamper mot Slovakia i går. «Som en mann som lagde en panne full av egggrønnsak mens han sto på scenen», sierer The Guardian.

Heia juksemakeren!
Han har forsøkt å skjule hellige jøkav. Likevel er det all grunn til å juble vilt for Julia Stepanova (29) den som har fått løpe i Rio de Janeiro.

Slik ble toppreneren dopingatt
Politiet hadde rapport på den profilerte friidrettsreneren Jana Aden (50) døgnet først i fire uker, før han ble arrestert av maskerte bewegelsesmedlemmer i går.

Norske bønder skyr EU som pesten - britiske bønder er ihuga EU-tilhengere
SUGDEN/ENGLAND/DEBBERG (VG): I Storbritannia er bøndene værre forkjempere for EU. I Norge er de stort i mot. De forstår hverandre godt.

Kanye West produserer debutalbumet til gaterapper
Cameron Grey Improviserte foran rapperen utenfor House of Blues i Los Angeles i 2015. Nå er den første videoen til det norske talentet klar.

Nettsideicon

Til nettsider kan man koble et icon som skal vises i faner, adressefelt og bokmerker. Dette iconet gjør fanen både mer informativ og tiltalende enn det opprinnelige «arket».



Å sette inn et slik icon er svært enkel. Det eneste du trenger gjøre er å plassere inn følgende kode i **<head>**-delen av nettsiden:

```
<link rel="shortcut icon" href="nettside.ico" />
```

Det mest kompliserte er å lage fila *nettside.ico*. Denne fila skal være et bilde på minst 60 x 60 (selv om helt opp til 192 x 192 er i vanlig bruk) piksler. De aller fleste bilderedigeringsprogram kan eksportere en slik fil ganske lett, men det å tegne et bra icon som ser bra ut i en visning på 60 x 60 piksler er en utfordring. Her finnes det imidlertid en god del ferdig icon på nett, men pass på så du velger et som ikke er opphavsrettsbeskyttet.



8 Hurtigreferanse HTML/CSS

HTML

- Grunnstruktur**
- Formatering av tekst**
- Escapetegn**
- Bilder**
 - Sette inn en figur
 - Skalere bilder med HTML
- Linker**
- Tabeller**
- Lister**
 - Uordnet liste
 - Ordnet liste
 - Nestet liste
 - Definisjonslister

Medieinnhold

- Lyd
- Video

Semantiske tagger

Nettsideicon

CSS

Kobling mellom HTML og CSS

Grunnstruktur

Box model

- Kantlinjer
- Marger
- Sentrering
- Skygge
- Størrelse og overflow
- Visningsmetoder

Selectorer

- Attributtverdier
- Pseudoelementer
- Pseudoklasser

Måleenheter

- Absolutte
- Relative

Fargeangivelser

Justere tekst

- Fonter
- Webfonter

CSS Bilder

- Tabeller

Posisjonering

- Relativ
- Fixed
- Absolute
- Lagjustering
- Floating

Flexbox

Medietyper

Media queries

- Skalere bilder med CSS og media queries

HTML

Grunnstruktur

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="stilarkfil.css" />
  </head>

  <body>
    <p>Innhold</p>
  </body>
</html>
```

Formatering av tekst

<code><p>...</p></code>	Lager paragraf
<code><pre>...</pre></code>	Lager en preformatert paragraf
<code><h1>...</h1></code>	Største overskrift
<code><h6>...</h6></code>	Minste overskrift
<code>...</code>	Markerer viktig tekst - Endrer mening
<code>...</code>	Markering av viktig tekst - Endrer ikke mening
<code><mark>...</mark></code>	Markerer relevant del av tekst for leseren

Escapetegn

<	<code>&lt;</code>
>	<code>&gt;</code>
&	<code>&amp;</code>
mellomrom	<code>&nbsp;</code>
"	<code>&quot;</code>
©	<code>&copy;</code>

Bilder

Sette inn et bilde:

```

```

Sette inn bilde med størrelse angitt i HTML:

```

```

Sette inn en figur:

```
<figure>
    
    
    <figcaption>Bilde av giraffen vår og tvillingbroren hans</figcaption>
</figure>
```

Skalere bilder med HTML:

```
<picture>
    <source srcset="giraff-mobil.jpg" media="(max-width: 720px)">
    <source srcset="giraff-nettbrett.jpg" media="(max-width: 1280px)">
    <source srcset="giraff-desktop.jpg">
    
</picture>
```

Linker

Link til annen nettside:

```
<p>Gå til <a href="side2.html">neste side</a>.</p>
```

Link til annen nettside som åpnes i ny fane:

```
<p>Gå til <a href="side2.html" target="_blank">neste side</a>.</p>
```

Benytte et bilde som en link:

```
<a href="side2.html">
    
</a>
```

Linke til andre typer filer:

```
<p>Se vår <a href="manual.pdf">brukermanual</a>.</p>
```

Link til en e-postadresse:

```
<p>Send oss gjerne en <a href="mailto:kontakt@doemne.no">e-post</a>.</p>
```

Internlinker

Lage en markør for et en internlink:

```
<div id="kontaktinformasjon">
</div>
```

Linke til en internlink på samme nettside:

```
<a href="#kontaktinformasjon">Gå til kontaktinformasjon</a>
```

Linke til en internlink på en annen nettside:

```
<a href="omoss.html#kontaktinformasjon">Gå til kontaktinformasjon</a>
```

Tabeller:

<code><table>...</table></code>	Omslutter tabellen
<code><tr>...</tr></code>	Omslutter en tabellrad
<code><td>...</td></code>	Omslutter en tabellcelle (data)
<code><th>...</th></code>	Omslutter en tabellcelle (header)

Enkel struktur:

```
<table>
  <tr><td>Deltaker 1</td><td>50</td><td>75</td></tr>
  <tr><td>Deltaker 2</td><td>50</td><td>75</td></tr>
</table>
```

Bruk av tabellheadinger:

```
<table>
  <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
  <tr><th>Ole Olsen</th><td>32</td><td>64</td></tr>
  <tr><th>Per Persen</th><td>56</td><td>75</td></tr>
</table>
```

Tabellceller som dekker flere rader/kolonner:

```
<table>
  <tr><th>Navn</th><th>Konkurranse 1</th><th>Konkurranse2</th></tr>
  <tr><th>Lise Nilsen</th><td colspan="2">Diskvalifisert</td></tr>
  <tr><th rowspan="2">Trude Hansen</th><td>23</td><td>44</td></tr>
  <tr><td>93</td><td>16</td></tr>
</table>
```

Lister:

<code>...</code>	Markerer en uordnet liste
<code>...</code>	Markerer en ordnet liste
<code>...</code>	Markerer et listepunkt
<code><dl>...</dl></code>	Markerer en definisjonsliste
<code><dt>...</dt></code>	Markerer ordet som skal definisieres
<code><dd>...</dd></code>	Markerer beskrivelsen av ordet som defineres

KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Ordnet liste:

```
<ul>
    <li>Blå</li>
    <li>Gul</li>
    <li>Grønn</li>
</ul>
```

Ordnet liste:

```
<ol>
    <li>Blå</li>
    <li>Gul</li>
    <li>Grønn</li>
</ol>
```

Nestet liste:

```
<ul>
    <li>Blå</li>
    <li>Gul
        <ul>
            <li>Lys</li>
            <li>Mørk</li>
        </ul>
    </li>
    <li>Grønn</li>
</ul>
```

Definisjonslister:

```
<dl>
    <dt>HTML</dt>
    <dd>Språk for å beskrive innhold og struktur i en nettside</dd>
    <dt>CSS</dt>
    <dd>Språk for å beskrive utseende til en nettside</dd>
</dl>
```

Medieinnhold

Lyd

```
<audio controls="controls">
    <source src="lydfil.mp3" type="audio/mpeg">
    <source src="lydfil.ogg" type="audio/ogg">
    Nettleseren støtter ikke lydavspilling
</audio>
```

Attributter

controls="controls"	Angir at vi ønsker ha avspillingskontrollere
autoplay="autoplay"	Angir at vi ønsker at lydavspillingen skal starte så snart nettsiden er lastet
loop="loop"	Angir at vi ønsker at lydavspillingen skal starte på nytt når den kommer til slutten

Video

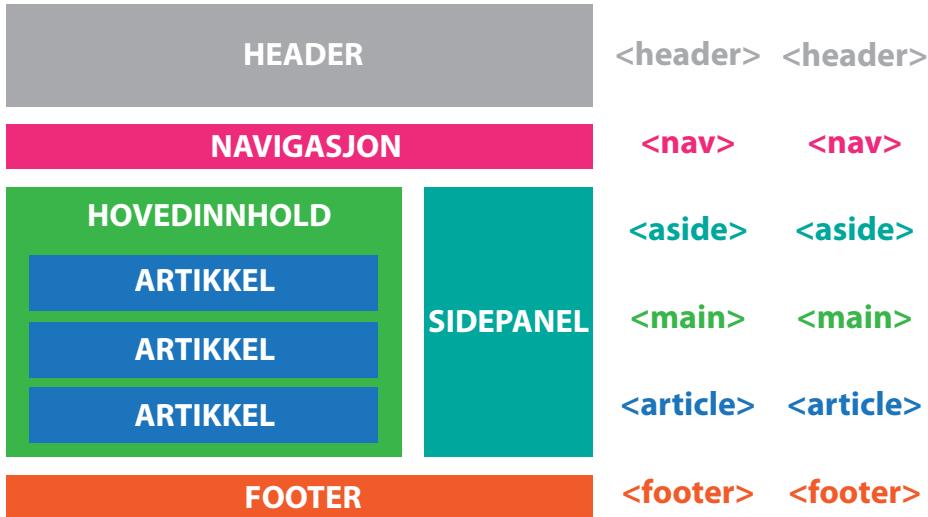
```
<video width="640" height="480" controls="controls">
  <source src="videofil.mp4" type="video/mp4">
  <source src="videofil.ogg" type="video/ogg">
  Nettleseren støtter ikke videoavspilling
</video>
```

Attributter

controls="controls"	Angir om vi ønsker ha avspillingskontrollere
autoplay="autoplay"	Angir at vi ønsker at videoavspillingen skal starte så snart nettsiden er lastet
loop="loop"	Angir at vi ønsker at videoavspillingen skal starte på nytt når den kommer til slutten
poster="url"	Spesifiserer et bilde som skal vises før videoen er lastet/starter. Vil også bli benyttet ved utskrift



Semantiske tagger



- <main> Hovedinnholdet på siden
- <nav> Navigasjonsmenyen på siden
- <header> Headeren på nettsiden
- <footer> Footeren på nettsiden
- <section> En del av nettsiden eller en del av en <article>
- <article> Selvstendig blokk med innhold
- <aside> Innhold som er "på siden av" hovedinnholdet

```
<time datetime="2016-12-14T09:30:15">neste time</time>
<abbr title="Norsk rikskringkasting">NRK</abbr>
```

Nettsideicon

```
<link rel="shortcut icon" href="nettside.ico" />
```

CSS

Skrivemåten **egenskapsnavn: verdi1 | verdi2 | verdi3;** som er tatt i bruk enkelte steder i denne seksjonen betyr at én av de ulike verdiene skal benyttes.

Kobling mellom HTML og CSS

Angi id:

```
<p id="ingress">Tekst</p>
```

Angi klasse:

```
<p class="brodtekst">Tekst</p>
```

Angi flere klasser (multiple class):

```
<p class="infoboks uthevet viktig">Tekst</p>
```

Taggen **<div>** - Markere en del av nettsiden:

```
<div id="brodtekst">
    <p>Første avsnitt</p>
    <p>Andre avsnitt</p>
    <p>Tredje avsnitt</p>
</div>
```

Taggen **** - Markere en del av innholdet i en tagg:

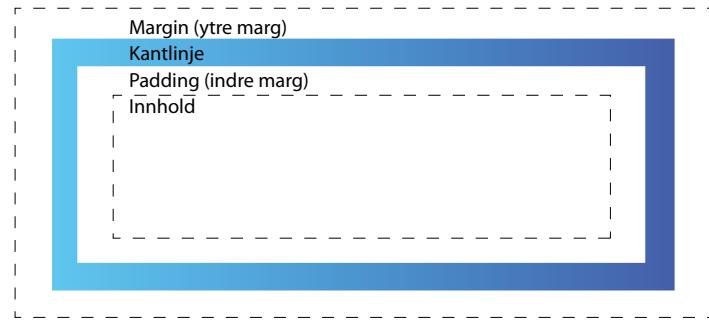
```
<p>Ole fikk <span class="tall">5</span> poeng og Per fikk <span class="tall">10</span> poeng.</p>
```



Grunnstruktur

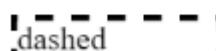
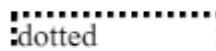
```
selector {
    egenskap: verdi;
    egenskap: verdi;
    egenskap: verdi;
}
```

Box model



Kantlinjer

```
border-color: red;
border-radius: 10px;
border-width: 1px;
border-style: solid;
```



Ulike verdier for kantlinje:



KOPIERING ER TILLATT FOR BRUK I SKOLEÅRET 2016/2017. DERETTER ER KOPIERING ULOVLIG.

Marger

```
margin: 10px;  
margin-top: 10px;  
margin-left: 5px;  
margin: 10px, 5px, 10px, 5px;
```

Sentrering

```
#bilde {  
    width: 300px;  
    margin-left: auto;  
    margin-right: auto;  
}
```

Skygge

```
box-shadow: 15px 25px;  
box-shadow: 15px 25px 10px 15px gray;
```

Størrelse og overflow

```
p {  
    width: 400px;  
    min-height: 400px;  
    max-height: 800px;  
    overflow-y: auto; /* Har også verdien scroll og none og kan settes  
                      som overflow-x, overflow-y og bare overflow */  
}
```

Visningsmetoder

```
display: block | inline | inline-block;
```

Selectorer

A og B skal erstattes med navnet på tagger eller andre selectorer som id- eller klasseangivelse

- A** Alle forekomster av taggen <A>
- #A** Taggen med id A
- .A** Alle tagger med klassen A
- A.B** Talle forekomster av taggen <A> som tilhører klassen B
- A, B, C** Alle forekomster av taggene <A>, og <C>
- A B** Alle forekomster av taggen inni taggen <A>
- *** Alle tagger
- A > B** Alle forekomster av taggen direkte inni taggen <A>
- A + B** Alle forekomster av taggen etter første forekomst av taggen <A>

Attributtverdier

- A[B]** Alle forekomster av taggen <A> der attributtet *B* er satt
- A[B^="C"]** Alle forekomster av taggen <A> der attributtet *B* er satt og dette starter med teksten *C*
- A[B\$="C"]** Alle forekomster av taggen <A> der attributtet *B* er satt og dette slutter med teksten *C*
- A[B=="C"]** Alle forekomster av taggen <A> der attributtet *B* er satt og dette er lik teksten *C*
- A[B*="C"]** Alle forekomster av taggen <A> der attributtet *B* er satt og dette inneholder teksten *C*

Pseudoelementer

::first-letter	Første bokstav
::first-line	Første linje
::before	Tekst før elementet
::after	Tekst etter elementet

Eksempel på tekst før og etter en tagg:

```
h2::before {
    content: "--";
}

h2::after {
    content: "--";
}
```

Pseudoklasser

:first-of-type	Første gang elementet dukker opp i nettsiden
:last-of-type	Siste gang elementet dukker opp i nettsiden
:only-of-type	Dersom elementet kun finnes én gang i nettsiden
:only-child	Dersom elementet er det eneste «barnet» inne i et annet element
:nth-child(x)	Angir barn nummer X under et element
:nth-last-child(x)	Angir barn nummer X (telt fra slutten) under et element
:nth-of-type(x)	Angir element nummer X av denne typen i hele nettsiden
:nth-last-of-type(x)	Angir element nummer X (telt bakfra) av denne typen i hele nettsiden

:first-child	Angir første barn under et element
:last-child	Angir siste barn under et element
:empty	Angir element uten innhold
a:visited	Link som er besøkt
a:link	Link som ennå ikke er besøkt
a:active	Linken i det vi klikker på den (og holder museknappen nede)
:hover	Elementet i det vi holder musepekeren over den

Måleenheter

Absolutive

- cm** Centimeter på visningsmediumet
- mm** Millimeter på visningsmediumet
- in** Tommer på visningsmediumet
- pt** Punkter på visningsmediumet. Det er 72 punkter i en tomme
- pc** Picas på visningsmediumet. Det er 12 punkter i en pica
- px** Måleenheten **px** er en litt rar, men veldig mye benyttet enhet. Den har ingen gitt størrelse, men avhenger av mediet. For medier med lav oppløsning er **1px** det samme som 1 piksel, men for medier med høy oppløsning er **1px** ofte flere piksler

Relative

- em** Angis i forhold til fontstørrelsen. Verdien **1.5em** tilsvarer f.eks. en og en halv gang fontstørrelsen
- %** Angis i forhold til foreldre-elementet. **100%** er samme størrelse som foreldre-elementet
- vw** Angis i forhold til *viewport* (området av nettsiden vi ser) sin bredde. Målet **100vw** tilsvarer viewport sin bredde. Målet **150vw** tilsvarer en og en halv gang viewport sin bredde. I en nettleser er viewport den synlige falten. Ved utskrift er viewport den delen av arket printeren kan skrive på
- vh** Tilsvarende som **vw**, men angis i forhold til viewport sin høyde

Fargeangivelser

```
color: red;                      /* Fargenavn */
color: rgb(75%,43%,100%);        /* RGB fra 0-100% */
color: rgb(191,110,255);         /* RGB fra 0-255 */
color: #bf6eff;                  /* Hexadesimalt */
color: rgba(191,110,255,0.6);    /* Med alpha (synlighet) fra 0-1 */
```

Justere tekst

```
font-style: italic | oblique | normal;
text-decoration: underline | overline | line-through | none;
font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900;
font-size: xx-small | x-small | small | large | x-large | xx-large | smaller |
larger | medium | måelenhet;
line-height: måleenhet;
text-align: center | left | right | justify;
text-transform: uppercase | lowercase | capitalize;
```

Fonter

```
font-family: "Times New Roman";
font-family: "Times New Roman", Georgia, Serif;
```

Generiske fonter: **monospace**, **fantasy**, **serif**, **sans-serif** og **cursive**

De vanligste fontene i bruk:

Arial, Helvetica, sans-serif

Arial Black, Gadget, sans-serif

Comic Sans MS, cursive, sans-serif

Courier New, Courier, monospace

Georgia, serif

Impact, Charcoal, sans-serif

Lucida Console, Monaco, monospace

Lucida Sans Unicode, Lucida Grande, sans-serif

Palatino Linotype, Book Antiqua, Palatino, serif

Tahoma, Geneva, sans-serif

Times New Roman, Times, serif

Trebuchet MS, Helvetica, sans-serif

Verdana, Geneva, sans-serif

Webfonter

```
@font-face {
    font-family: Superfont;
    src: url("superfont.ttf");
}

p {
    font-family: Superfont, sans-serif;
}
```

CSS-bilder

```
<div id="bilde">
</div>
```

```
#bilde {
    width: 200px;
    height: 200px;
    background-image: url("bilde.jpg");
}
```

background-repeat Dersom elementet er sørre en bildet, skal det da repetere (**repeat**) , repetere horisontalt (**repeat-x**) , repetere vertikalt (**repeat-y**) eller ikke repetere (**no-repeat**)

background-attachment Skal bakgrunnen scrollle sammen med nettsiden (**scroll**) eller stå fast selv om bakgrunnen scroller (**fixed**)

background-position En angivelse av posisjonering i horisontal og vertikal retning ved hjelp av nøkkelordene **left / right / center** og **top / bottom / center**

background-size Her angir vi en størrelse for bredde og høyde på bakgrunnsbildet

opacity En tallverdi som angir fra 1 (ikke transparent) til 0 (helt usynlig) hvor gjennomsiktig bakgrunnen skal være

Tabeller

```
table, th, td {
    border: 1px solid black;
}
```

Posisjonering

Relativ

```
h1 {
    position: relative;
    left: 50px;
}
```

Fixed

```
#banner {
    position:fixed;
    top: 0;
    left: 0;
    width: 100%;
}
```

Absolute

```
#banner {
    position:absolute;
    top: 0;
    left: 0;
    width: 100%;
}
```

Lagjustering

`z-index: 5;`

Positive tall: Fremover i nettsiden

0: Originalposisjon

Negative tall: Bakover i nettsiden

Floating

```
img {
    float: left;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla
 egestas lectus et velit luctus, eu dictum tortor luctus. Aenean
 at aliquet arcu. Nullam eget massa ut magna interdum viverra.
 Aliquam tincidunt enim sit amet magna molestie malesuada. Ut
 aliquam turpis massa, sit amet consequat tellus suscipit vitae.
 Phasellus eu dignissim velit. Ut consequat vel elit in
 condimentum. Donec vitae elit porttitor, varius turpis sit amet,
 cursus lorem. Pellentesque vel vulputate nisi. Aenean eros
 lacus, posuere a varius id, bibendum rutrum mauris. Nam
 feugiat, felis eget congue ultrices, elit metus accumsan ex, nec
 commodo leo magna nec enim. Proin facilisis vulputate tellus non tincidunt.
 Vestibulum dictum facilisis metus.

Flexbox

Oppsett:

```
<div id="wrapper">
  <div id="boks1">Boks 1</div>
  <div id="boks2">Boks 2</div>
  <div id="boks3">Boks 3</div>
</div>
```

Egenskaper for wrapperen:

```
#wrapper {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  flex-wrap: wrap
  align-content: space-between;
  align-content: stretch;
  align-items: stretch;
}
```

Verdier for **justify-content** (og **align-content**):

center	sentrerer elementene på midten av nettsiden
flex-start	justerer elementene til venstre (ved row) eller topp (ved column)
flex-end	justerer elementene til høyre (ved row) eller bunn (ved column)
space-between	fordeler ledig plass mellom elementene
space-around	fordeler ledig plass mellom, før og etter elementene

Egenskaper for elementene (*flexy*):

```
#boks1 {
  flex-grow: 1;
  flex-shrink: 1;
  flex-basis: 300px;
  height: 50px;
  order: 3;
}
```

Egenskaper for elementene (*fixed*):

```
#boks2 {
  height: 50px;
  width: 50px;
  order: 2;
}
```

Medietyper

Helt ulike stilark for skjerm og utskrift:

```
<link rel="stylesheet" type="text/css" media="screen" href="skjerm.css" />
<link rel="stylesheet" type="text/css" media="print" href="utskrift.css" />
```

Stilark der vi spesifiserer endringer for utskrift:

```
<link rel="stylesheet" type="text/css" media="all" href="stil.css" />
<link rel="stylesheet" type="text/css" media="print" href="utskrift.css" />
```

I samme stilark:

```
p {
    color: red;
}

h1 {
    color: green;
}

@media print {
    p {
        color: blue;
    }
}
```

Media queries

I samme stilark:

```
body {
    color: black;
}

@media all and (min-width: 500px) {
    body {
        color: red
    }
}
```

I ulike stilark:

```
<link rel="stylesheet" media="all and (orientation:portrait)" href="stil.staaende.css" /> 
<link rel="stylesheet" media="all and (orientation:landscape)" href="stil.liggende.css" /> 
```

Egenskaper:

width	Bredde på skjermflate
height	Høyde på skjermflate
device-width	Bredde på enheten
device-height	Høyde på enheten
orientation	Om enheten er i landscape eller portrait modus (gjelder både skjerm og papir)
aspect-ratio	Forhåndstallet i bredde/høyde for skjermflaten. Uttrykkes som f.eks. 16/9
device-aspect-ratio	Forhåndstallet i størrelse for enheten
color-index	Antall ulike farger som kan vises. f.eks. 256
monochrome	Antall gråtoner om sort/hvitt enhet. Ellers 0
resolution	Oppløsningen på enheten, slik som 300dpi

Kode for at mobilskalering skal fungere på skjerm:

```
<meta name="viewport" content="width=device-width, initial-scale=1;" />
```

Skalere bilder med CSS og media queries

```
#bilde {  
    width: 500px;  
    background-image: url("giraff-dekstop.jpg");  
}  
  
@media all and (max-width: 768px) {  
    #bilde {  
        background-image: url("giraff-mobil.jpg");  
    }  
}  
  
@media all and (max-width: 1280px) {  
    #bilde {  
        background-image: url("giraff-nettbrett.jpg");  
    }  
}
```

9 Introduksjon til dynamiske nettsider

I dette kapitlet vil du lære

- om forskjellen mellom statiske og dynamiske nettsider
- om WampServer
- hvordan du installerer WampServer
- om mulige problemer ved bruk av WampServer
- om MAMP for Mac-brukere

Statiske og dynamiske nettsider

Tidligere var nettet i stor grad bestående av såkalte *statiske nettsider*. Dette er nettsider som alltid vil ha samme innhold, om ikke eieren av nettsiden endrer den. Denne formen for nettsider egner seg greit til mindre webområder som kun er ment for å presentere informasjon der innholdet stort sett er konstant/statisk.

I den senere tid har man imidlertid sett at statiske nettsider som oftest kommer til kort med hva man ønsker at en nettside skal benyttes til. Derfor blir nå en stadig større andel av nettsidene *dynamiske*.

Dynamiske nettsider lar oss lage nettsider som er lette å oppdatere, og som kan koble sammen utseende, struktur og innhold fra ulike kilder idet brukeren besøker nettsidene. Dette kjenner vi igjen fra nettbutikker, nettaviser, webkameraer, blogger, sosiale nettverk osv.

I tillegg til denne enklere oppdateringen av nettsidene gir også teknikken bak dynamiske nettsider brukeren mulighet til å styre nettsidene. Dette gjør at vi f.eks. kan plassere produkter i en handlekurv, skrive kommentarer på en nyhetsartikkel, søke i søkmotorer eller legge nye venner til vår konto på f.eks. Facebook.

Selv om det er viktig å skille begrepene statiske og dynamiske nettsider, er faktisk ikke teknologiene så forskjellige. Mens statiske nettsider utelukkende består av HTML-koder, består de dynamiske nettsidene av en blanding mellom HTML-koder og programkode.

Det er disse programkodene (f.eks. C#, PHP, Java, JavaScript osv.) som genererer den ferdige nettsiden. Faktisk er resultatet av programkoden i en dynamisk nettside som genereres på webserveren, en temporær statisk nettside som består av utelukkende HTML-koder.

Det eneste som kompliserer teknikken rundt dynamiske nettsider, er at programkoden et eller annet sted må bli utført. Her finnes det to mulige løsninger, nemlig at programkoden utføres i nettleseren (*client-side scripting*) slik som f.eks. JavaScript, eller at programkoden utføres i webserveren (*server-side scripting*) slik som f.eks. PHP og C#.

I denne boka skal vi fokusere på server-side scripting via PHP. Dette medfører at hver gang en nettleser spør webserveren om å få se en nettside, vil webserveren fullføre nettsiden gjennom å kjøre programkoden og så gi HTML-koden tilbake til nettleseren.

Statistiske websider:



Dynamiske websider:



Så lenge vi har nettsidene våre på en ekstern webserver, har vi lite bekymringer med oppsettet av dette systemet. Dersom vi derimot skal utvikle nettsidene lokalt på vår maskin, slik vi til nå har gjort i denne boka, må vi installere en såkalt lokal webserver. En lokal webserver er faktisk ikke noe annet enn det samme produktet som kjører på andre webservere vi kjenner. Den eneste forskjellen er at vi som oftest ikke har noe domenenavn (slik som f.eks. www.vg.no) som identifiserer webserveren/nettsidene.

I denne boka skal vi benytte *Apache* som webserver. Dette er et open source-prosjekt og er derfor fritt tilgjengelig. Apache er også forholdsvis lett å installere, samtidig som den støtter de fleste teknologier slik som PHP, som vi skal benytte.

Det er viktig å alt nå påpeke at denne boka ikke er ment som en opplæring i programmering. Vi kommer derimot til å gi deg et sett med ferdige kodeblokker som du kan benytte i dine egne prosjekter. Det er ikke å forvente at du skal forstå alle sider med disse kodeblokkene nå. Det viktigste er at du forstå hvordan de skal benyttes.

Du vil finne en egen seksjon som inneholder en hurtigreferanse for disse kodeblokkene i denne boka. På bokas nettsider vil du kunne finne igjen både hurtigreferansen og en egen kodegenerator.

WampServer

I denne boka skal vi som sagt benytte *PHP* som serverspråk og *Apache* som webserver. I tillegg skal vi benytte *MySQL* som databasesystem. Dette er i utgangspunktet tre separate produkter, som vi vanligvis selv er ansvarlig for å få til å fungere sammen.

Ettersom disse tre produktene er så vanlige å benytte sammen, er det imidlertid mange som har laget såkalte *bundles* der produktene er satt sammen til ett. Vi skal benytte en slik bundle som kalles *WampServer* (**Windows-Apache-MySQL-PHP Server**).



WampServer er et produkt for Windows. Det finnes et tilsvarende produkt kalt *MAMP* for Mac. Vi skal beskrive MAMP til slutt i kapittelet. For Linux er Apache, MySQL og PHP applikasjoner som enkelt kan installeres gjennom operativsystemets pakkehåndteringsverktøy.

Installerer vi WampServer, har vi altså gjort hele jobben i én operasjon, og WampServer sørger selv for at de tre produktene fungerer sammen. WampServer tilbyr også et kontrollpanel for enkelt å styre alle de tre produktene.



Det finnes en også rekke andre slike bundler som også kan benyttes. XAMPP er et likeverdig produkt.

Installasjon av WampServer

Ettersom WampServer er fritt tilgjengelig og bygger på fritt tilgjengelig programvare, kan du laste ned programmet gratis fra denne siden:

<http://www.wampserver.com/en>

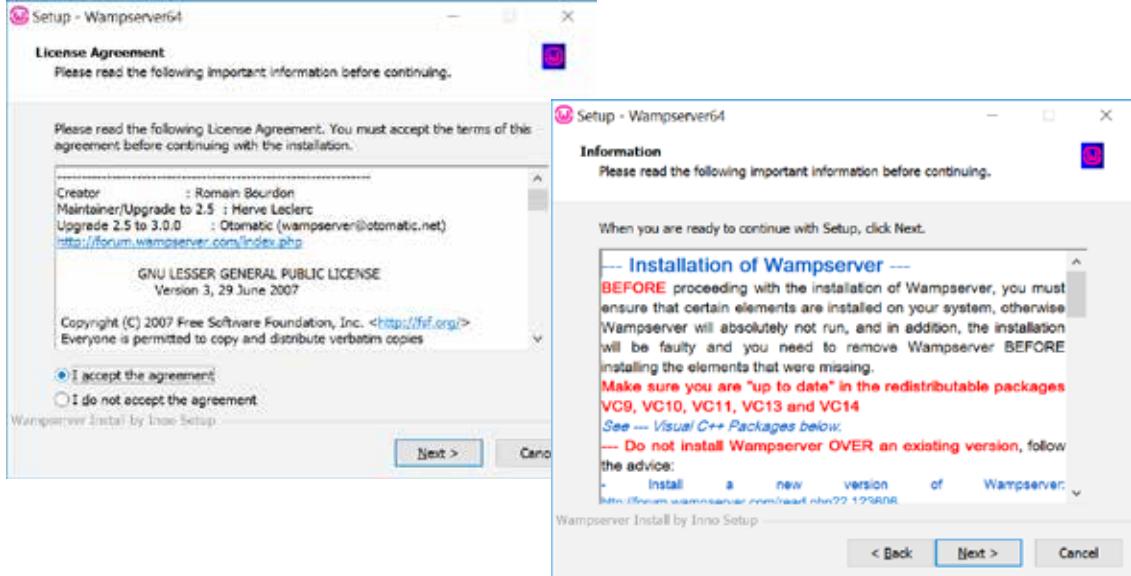
WampServer kommer stadig i nye versjoner, så denne installasjonsveiledningen kan avvike noe. Veileddingen er basert på versjon 3.0.

Dersom det kommer større endringer til WampServer i tiden fremover, vil vi legge ut oppdateringer på bokas nettsider.

Får du en feilmelding om en manglende DLL-fil (bl.a *VCRUNTIME140.dll*) under installasjonen må du installere *Visual C++ Redistributable for Visual Studio*. Denne kan du finne her: <https://www.microsoft.com/en-us/download/details.aspx?id=48145>



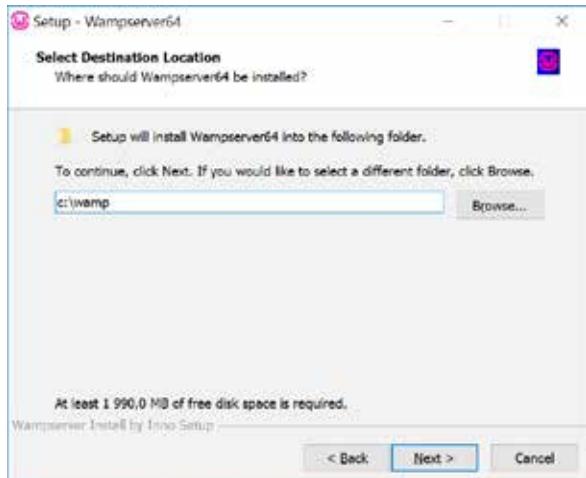
1. Følg så installasjonsveiledningen gjennom valg av språk, godkjenning av lisensavtalen og informasjon om installasjonen.



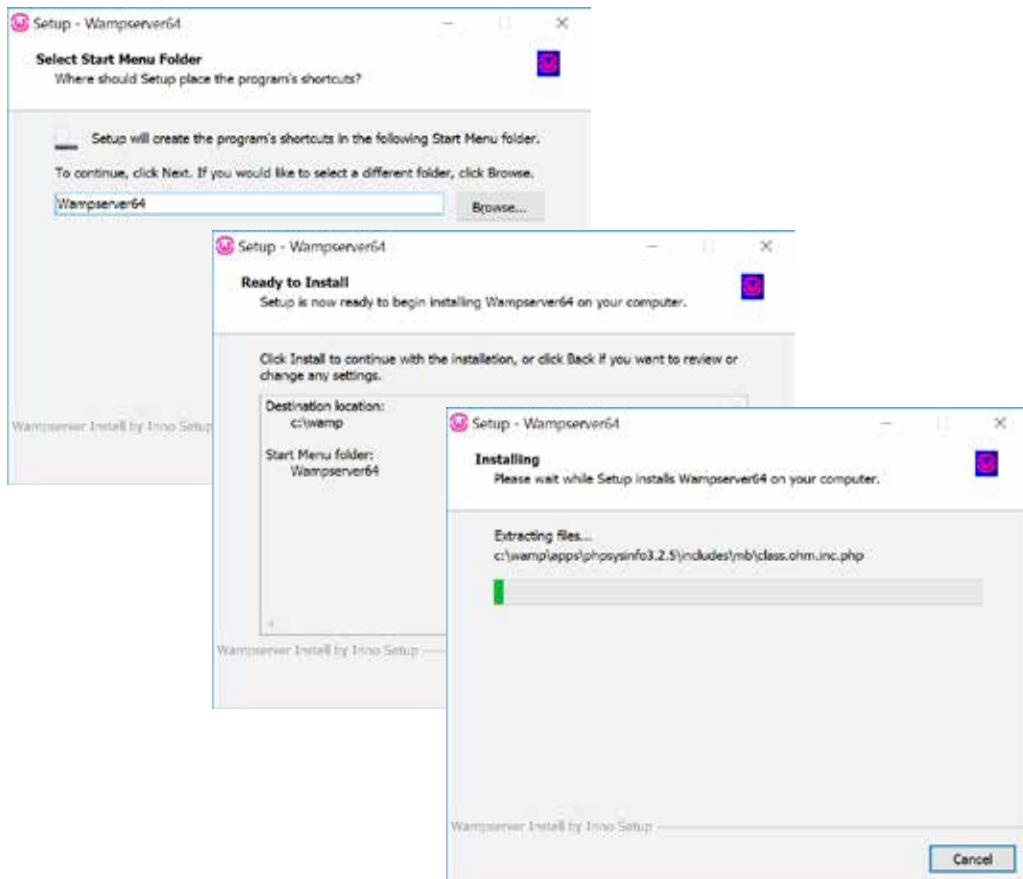
2. Du vil så få en forespørsel om hvor WampServer skal installeres. Det er en stor fordel om du plasserer WampServer i *c:\wamp*, slik som den foreslår. Om ikke dette er mulig, så kan du også velge alternative plasseringer, men vær forsiktig med mapper som er brukerstyrt og/eller virtuelle (slik som *Mine Dokumenter* osv.). Resten av denne boka vil ta utgangspunkt i at WampServer er installert i *c:\wamp*.



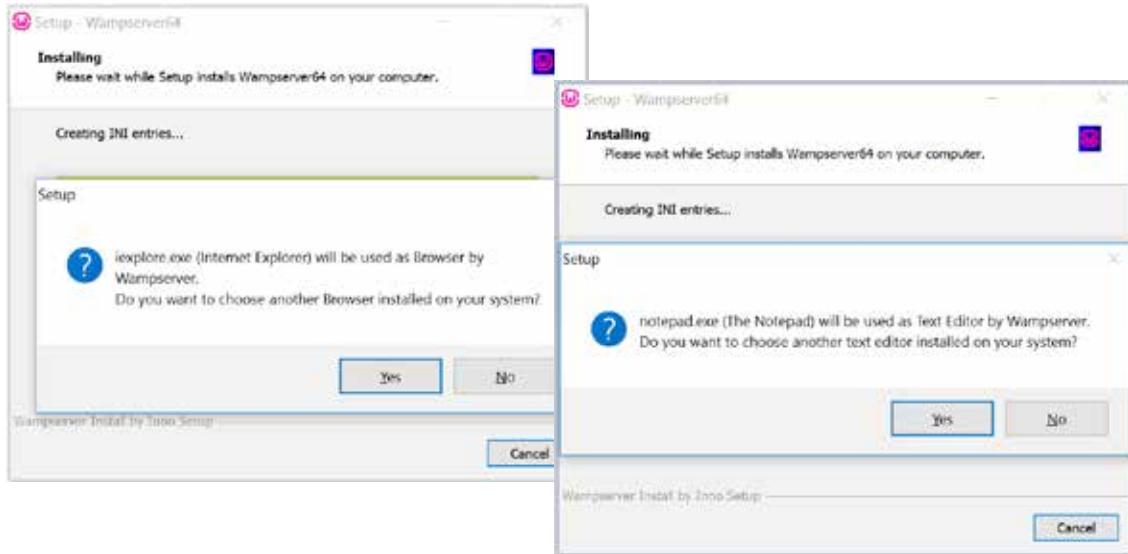
Om du installerer en 64-bit versjon vil den foreslå c:\wamp64. Velg allikevel c:\wamp som plassering for å få installasjonen mest mulig like resten av bokas fremgangsmåte.



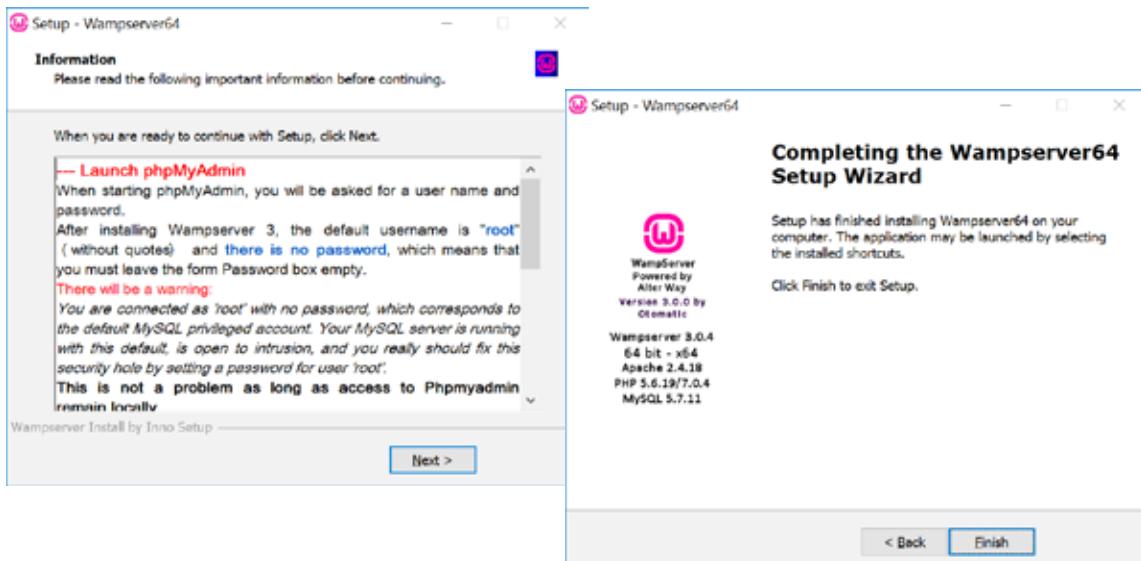
3. Fortsett så installasjonen gjennom godkjenningen av valgene og selve kopieringen av filer.



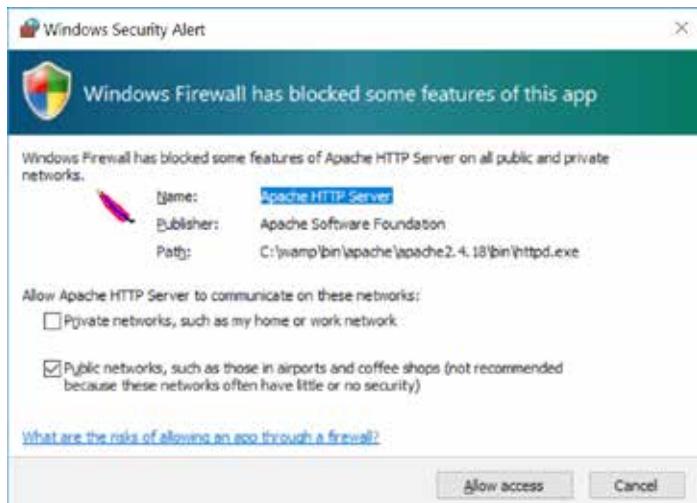
4. Når kopieringen av filer er fullført, vil installasjonsprosessen spørre deg om foretrukket nettleser og teskteditor. Her kan du velge **Yes** for å lte frem exe-filen til Chrome og Notepad++ eller du kan la det stå som det gjør ved å velge **No**.



5. Til slutt får du litt mer informasjon om installasjonen. Trykk **Next** gjennom de to skjermbildene.

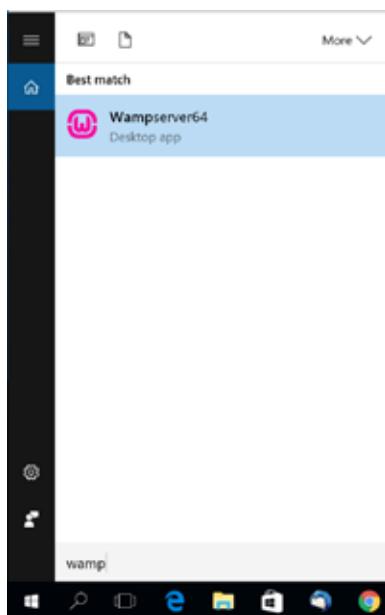


6. Får du spørsmål om å tillate Apache gjennom din brannmur, velger du at du ønsker å tillate dette (**Allow access**).

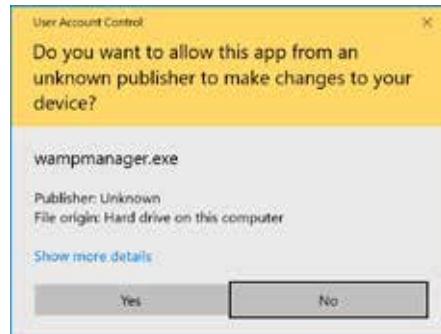


Starte WampServer

Etter å ha installert WampServer er det på tide å starte den. Dette gjøres ved å velge elementet *Wampserver/Wampserver64* i mappa *Wampserver/wampserver64* på startmenyen.

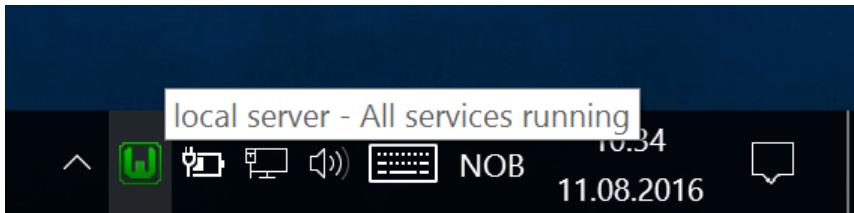


Du kan få en advarsel om at WampServer kommer til å endre visse deler av systemet. Dette er fordi WampServer må få tilgang til å registrere seg som en tjeneste. Dette må du tillate for at WampServer skal få starte.



Du vil nok bli ganske skuffet over effekten av å starte WampServer, for det skjer tilsynelatende svært lite. Det er fordi WampServer primært er en systemtjeneste til bruk for andre programmer og ikke en grafisk applikasjon som brukeren skal arbeide med. Derfor har WampServer et meget begrenset brukergrensesnitt.

Faktisk er det eneste du ser til WampServer, et lite **W**-ikon i *system tray* ved siden av klokka.



Litt ettersom hvor bra oppstarten av WampServer gikk, vil dette ikonet presentere en av følgende tre statuser:



Alt OK



En eller flere av tjenestene (Apache, MySQL) kunne ikke starte



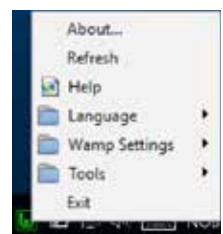
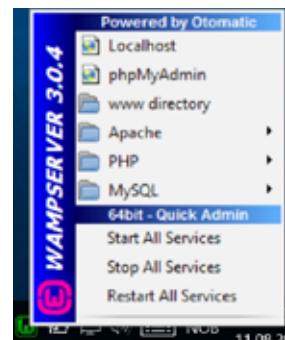
Tjenestene i WampServer er stoppet

Dersom alt gikk bra under oppstarten, kan du nå gjøre deg litt kjent med menyene til WampServer. Gikk det ikke bra vil neste seksjon forklare mulige feil.

Dersom du venstre klikker på ikonet til WampServer, får du opp en meny for administrasjon av tjenestene.

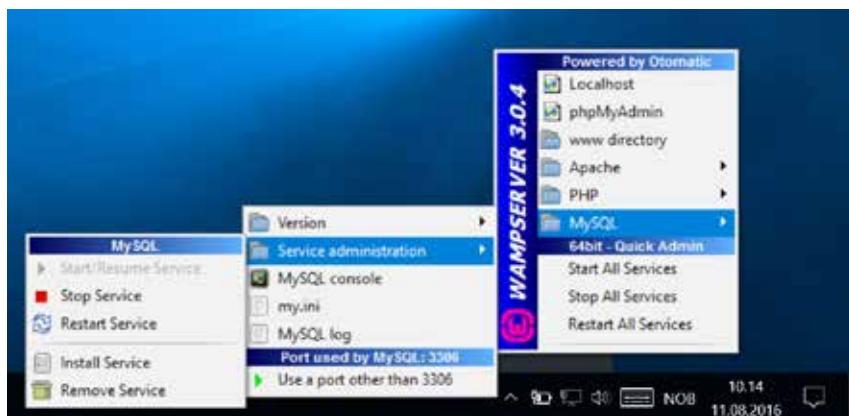
Her kan du også starte de ulike nettsidene WampServer tilbyr. I tillegg finnes det også en snarvei til webservermappa `c:\wamp\www`, gjennom menyvalget **www directory**.

Høyreklikker du på ikonet til WampServer, får du opp administrasjonen av selve WampServer. Det viktigste valget her vil være å avslutte WampServer gjennom **Exit**-kommandoen.

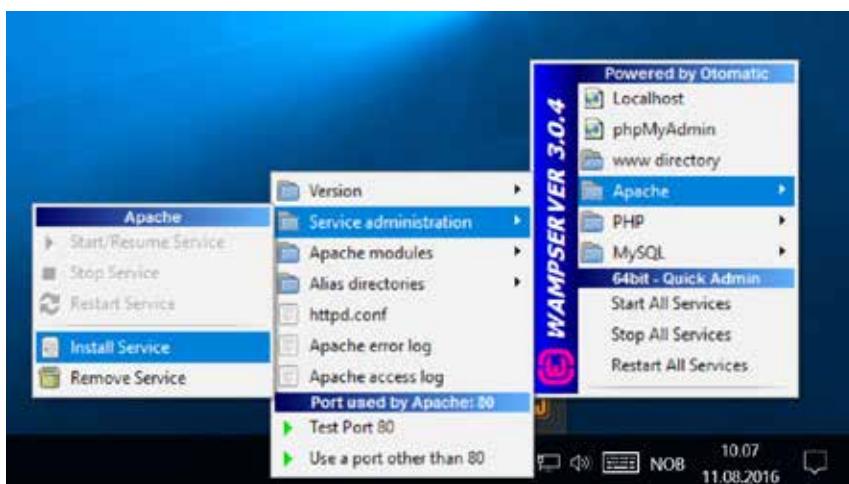


Feil under oppstart av WampServer

Du kan finne ut hvilken tjeneste som ikke starter ved å velge [tjenestenavn] > **Service administration**. Dersom du har mulighet til å stoppe tjenesten, er tjenesten nå kjørende og alt i orden.



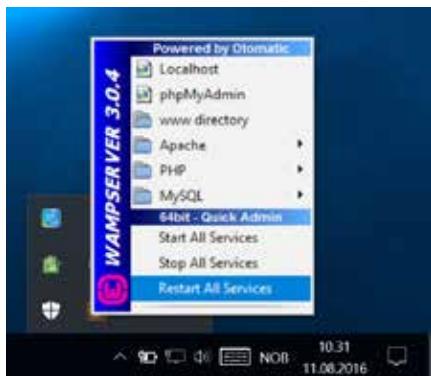
Dersom du har valget om å starte tjenesten aktivt får tjenesten ikke startet.



Er valgene grået ut må du installere tjenesten inn i selve operativsystemet, ved å velge **Install Service**.

TIPS PHP er ingen tjeneste, og kan derfor ikke startes eller stoppes. PHP er en integrert del av Apache.

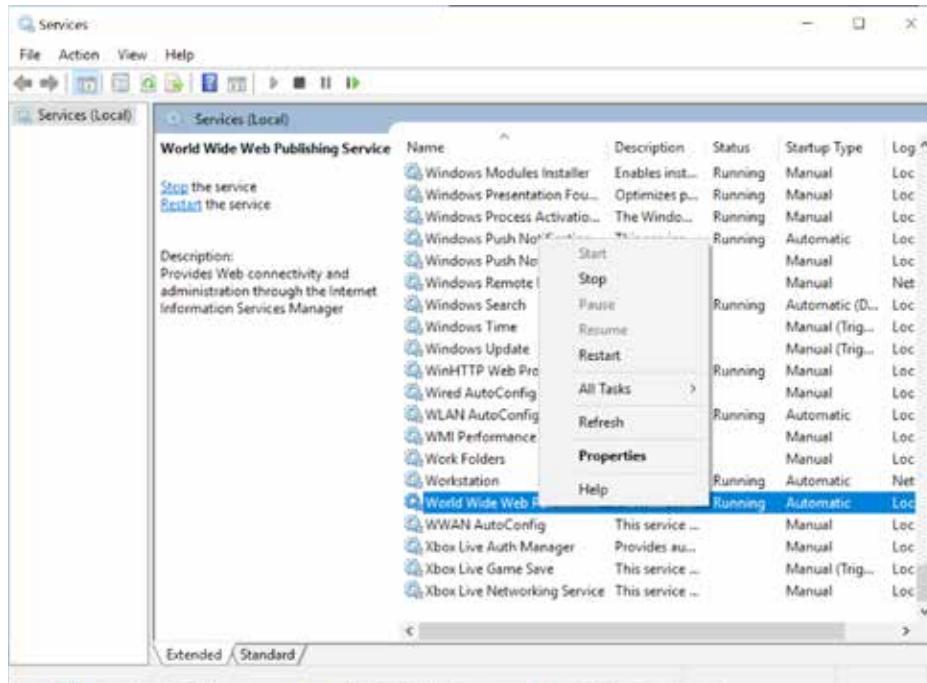
Dersom det skulle vise seg at en tjeneste ikke kunne starte, er den mest sannsynlige årsaken at du alt har noen lignende tjenester kjørende. Dette vil da vanligvis være en annen webserver eller en ekstra MySQL-installasjon. Disse tjenestene må da stoppes. Etter at tjenestene er stoppet, velger du å restarte WampServers tjenester på venstremenyen til ikonet for WampServer.



Avslutte IIS

Dersom webserveren *Microsoft Internet Information Services* (IIS) er installert, må du stoppe denne for å frigi port 80 til WampServer. Du kan med andre ord bare ha én webserver kjørende på maskinen om gangen, dersom du ikke tilordner ulike portnummer.

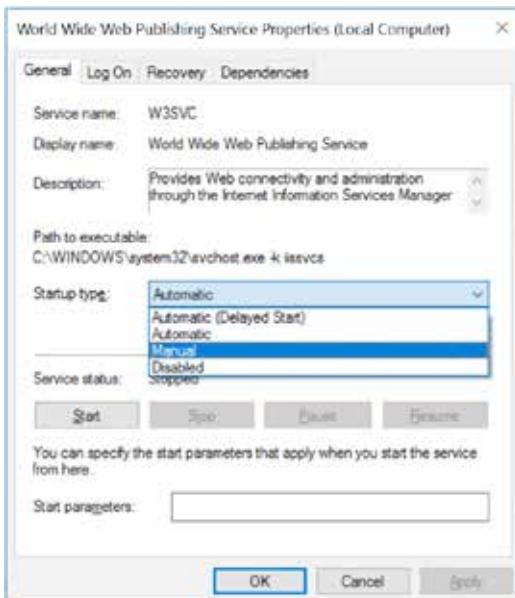
For å stoppe IIS går du til **kontrollpanel** og velger så **administrative verktøy**. Her velger du **Services/Tjenester**. I vinduet som kommer opp, leter du deg fram til elementet *World Wide Web Publishing Service/Webpubliseringstjeneste*, høyreklikker og velger **Stop**.





Elementet *World Wide Web Publishing Service* kan ha ulike navn i ulike versjoner av Windows.

Merk deg at denne framgangsmåten stopper IIS for denne sesjonen. Neste gang du starter maskinen, vil IIS igjen kjøre. For å stoppe IIS permanent velger du **Properties/Egenskaper** på høyremenyen og setter måten tjenesten skal starte på, til **Manuelt**.



Ved permanent stopping av IIS så vær klar over at IIS bl.a. vil kunne være påkrevd av ulike verktøy for intranett.

Avslutte andre MySQL-instanser

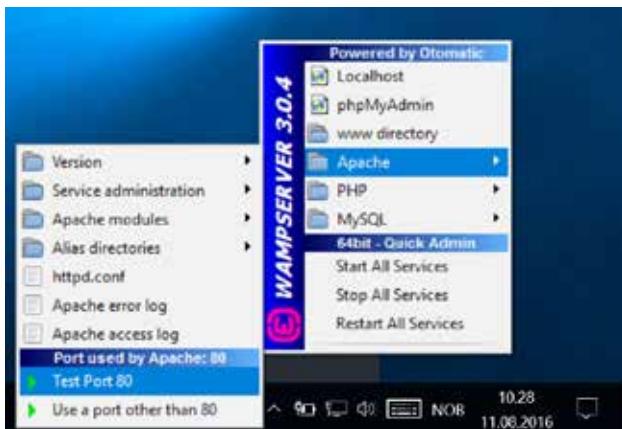
For å avslutte andre installasjoner av MySQL følger du samme framgangsmåte som for å avslutte IIS, men velger isteden tjenesten med navnet *MySQL*. Vær imidlertid klar over at eventuelle andre installasjoner av MySQL kan være i bruk av applikasjoner du benytter.

Andre applikasjoner som opptar port 80

Dersom fortsatt ikke WampServer vil starte, og evt. installasjoner av IIS og MySQL er stoppet, er årsaken vanligvis at andre applikasjoner oppter port 80.

Skype er kjent for å oppta port 80 og bør avsluttes før du starter WampServer, dersom installert. Pass på at du faktisk avslutter programmet, og ikke bare lukker det ned.

WampServer har også en test-funksjon der du kan finne ut hvilke programmer som skaper problemer. Dette finner du ved å velge menyen til **WampServer > Apache > Test Port 80.**



Et kommando-vindu forteller deg da hvilken tjeneste som opptar porten. Du må da avslutte disse før du forsøker starte tjenestene i WampServer på nytt.

```
c:\wamp\bin\php\php5.6.19\php.exe
***** Test which uses port 80 *****
===== Tested by command netstat filtered on port 80 =====

Test for TCP
Your port 80 is used by a processus with PID = 1780
The processus of PID 1780 is 'httpd.exe' Session: Services
The service of PID 1780 for 'httpd.exe' is 'wampapache64'
This service is from Wampserver - It is correct

Test for TCPV6
Your port 80 is used by a processus with PID = 1780
The processus of PID 1780 is 'httpd.exe' Session: Services
The service of PID 1780 for 'httpd.exe' is 'wampapache64'
This service is from Wampserver - It is correct

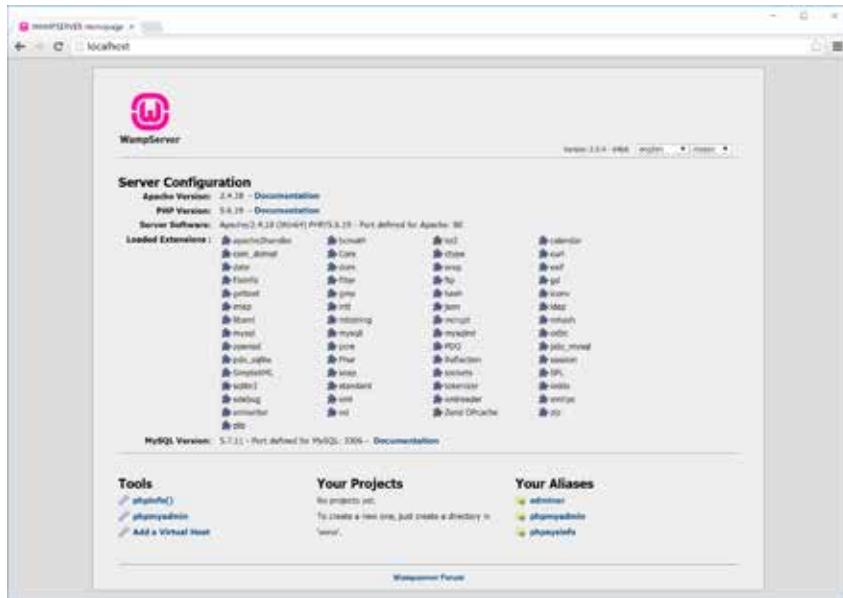
===== Tested by attempting to open a socket on port 80 =====

Your port 80 is actually used by :
Server: Apache/2.4.18 (Win64) PHP/5.6.19

--- Do you want to copy the results into Clipboard?
--- Type 'y' to confirm - Press ENTER to continue...
```

Teste WampServer

Etter at du er ferdig med å installere WampServer, og alle tjenester kjører, er det på tide å teste den. Dette kan enkelt gjøres ved å åpne WampServers startside i en nettleser. Skriv derfor inn adressen <http://localhost> i din nettleser, og kontroller at du får opp følgende nettside:



TIPS Du kan også velge menyelementet *localhost* på WampServer-menyen for å få opp denne startsiden.

MAMP

MAMP er et tilsvarende verktøy som WampServer for Mac. Dette finnes tilgjengelig for nedlasting her:

<http://www.mamp.info/en/>

Husk å velge gratisversjonen, og følg deretter installasjonsveiledningen.

Når du starter MAMP får du opp følgende skjermbilde:



MAMP har ikke alle mulighetene fra applikasjonen som WampServer har. Funksjonaliteten vi skal bruke videre i boka er imidlertid den samme.

Det er viktig å passe på at de to firkantene øverst til høyre har grønt fyll. Har de ikke det betyr det at respektive tjeneste ikke kunne starte. Du må da forsøke stoppe andre forekomster av webservere og MySQL-databaser.

Det er imidlertid mindre sannsynlig at du får en kollisjon i MAMP ettersom den kjører webserveren på port 8888 i stedet for port 80 som er standard. Dette medfører at alle henvisninger i denne boka til *localhost* i nettleserens adresselinje dermed må endres til *localhost:8888*.

Dette gjelder imidlertid ikke når vi senere kommer til å skrive database-tilkobling i PHP-koden, der vi fortsatt bruker *localhost*.



Skal vi f.eks. teste at webserveren kjører, gjøres dette ved å skrive følgende i nettleseren: <http://localhost:8888/MAMP/?language=English>

The screenshot shows the MAMP Pre-Release website for MAMP & MAMP PRO 4. The main header features a large white elephant logo on the left, the text "Pre-Release MAMP & MAMP PRO 4" in the center, and a large number "4" on the right. A "Free download" button is located below the main title. The page is divided into several sections:

- PHP**: Shows the current configuration of PHP. It includes a link to "phpinfo" which shows the current configuration of PHP.
- MySQL**: Shows connection parameters for MySQL. Host: localhost, Port: 8889, User: root, Password: root, Socket: /Applications/MAMP/tmp/mysql/mysql.sock.
- MAMP Version**: Shows the current version is 3.5, with a link to "Update (3.5.2) available!"
- News**: Lists recent news items:
 - MAMP & MAMP PRO 3.5.1 (OS X) ready for download
 - July, 2016 - We've updated MAMP & MAMP PRO to Version 3.5.1
 - MAMP & MAMP PRO supports php 7.0.8
 - MAMP & MAMP PRO 4 pre-release
 - July, 2016 - MAMP & MAMP PRO 4 pre-release
 - No available for free download
 - appelutx - The MAMP company - is looking for a Windows Developer
 - Full time permanent role for the further development of MAMP and MAMP PRO in our German Office.
 - MAMP & MAMP PRO 3.5 (OS X) ready for download
 - December, 2015 - We've updated MAMP & MAMP PRO to Version 3.5
 - MAMP & MAMP PRO supports php 7
 - MAMP PRO for Windows now on sale
 - August, 2015 - MAMP PRO for Windows now on sale
- Examples**: Shows examples for different programming languages: PHP < 5.6.x, PHP >= 5.6.x, Python, Perl. It also shows a code snippet for MySQL connection: \$user = 'root'; \$password = 'root';

Katalogen som tilsvarer *c:\wamp\www* heter med MAMP */Applications/MAMP/htdocs*. Alle henvisninger i boka til *c:\wamp\www* må derfor av MAMP-brukere erstattes med *Applications/MAMP/htdocs*.

10 Din første dynamiske nettside

I dette kapitlet vil du lære

- å lage en enkel dynamisk nettside
- hvordan du tester en dynamisk nettside
- om nettressurser for PHP

I forrige kapittel installerte vi webserveren vi trenger, og gjorde alt klart. Nå skal vi gå i gang med å lage vår første dynamiske nettside. Fremgangsmåten for hvordan du lagrer og hvordan du tester denne nettsiden vil gjelde alle prosjekter som innebærer dynamiske nettsider.

Opprette den dynamiske nettsiden

For å komme i gang skal vi lage en veldig enkel nettside som viser dagens dato.



Vi har endret fargekoding for PHP-tektstregner (det mellom " " og ") i forhold til Notepad++. Vi benytter her orange i stedet for grått for å få det mer synlig på trykk.

1. Åpne mal-filen som du laget i første del av denne boka i din kodeeditor. Du kan fjerne referansen til stilarket, da vi ikke skal lage nettsider med noe spesielt design nå. Lagre gjerne denne utgaven av malen som *mal.php* for videre bruk i denne delen av boka
2. Legg inn følgende HTML:

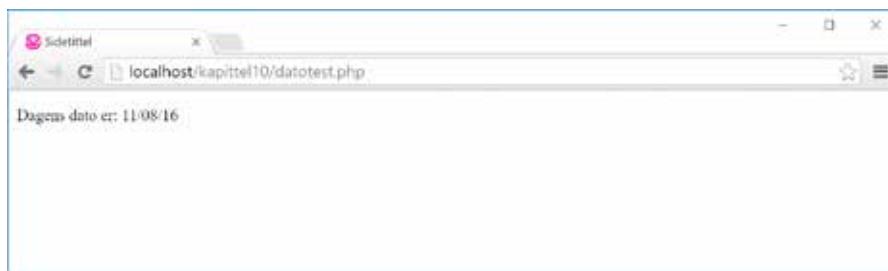
```
<!DOCTYPE html>
<html>
    <head>
        <title>Sidetittel</title>
        <meta charset="utf-8" />
    </head>

    <body>
        <p>Dagens dato er: </p>
    </body>
</html>
```

3. Skriv så inn PHP-koden som produserer dagens dato på posisjonen der vi ønsker at resultatet skal stå:

```
<body>
    <p>Dagens dato er: <?php echo date("d/m/y"); ?></p>
</body>
```

4. Velg nå å lage en ny mappe kalt *kapittel10* i mappen *c:\wamp\www* (*/Applications/MAMP/htdocs* for MAMP-brukere)
5. Lagre fila som *datotest.php* i mappa vi nettopp opprettet
6. Åpne nettleseren og skriv inn *http://localhost/kapittel10/datotest.php* i adressefeltet (*http://localhost:8888/kapittel10/datotest.php* for MAMP-brukere). Du skal nå få opp en enkel nettside som viser dagens dato



7. Velg nå å vise kildekoden bak nettsiden (**Ctrl + U** i de fleste nettlesere). Som du ser har webserveren erstattet PHP-koden med dagens dato også i kildekodene. Det er altså ikke nettleseren, men webserveren som produserer denne informasjonen



Merk deg at det å lage en slik nettside uten PHP, eller en annen teknologi for dynamiske nettsider, ville være praktisk umulig. I så fall måtte man manuelt oppdatert den statiske nettsiden hver eneste dag.

Ting å huske på når du lager dynamiske nettsider

Det er noen svært viktige ting du må huske på når du arbeider med dynamiske nettsider:

- Filendelsen på PHP-filene til en dynamisk nettside må alltid være *php*. Vi kunne ikke lagret denne fila som *datotest.html*. Hadde vi gjort det ville ikke koden blitt utført av webserveren, og siden nettleseren ikke forstår PHP-koder ville de stått igjen som en uleselig del av nettsiden. Vi ville da sett PHP-kodene i visningen av kildekoden, og ikke sett noe resultat av kodene i visningen av selve nettsiden.
- De dynamiske nettsidene må alltid plasseres i webserverens spesielle mappe for web-relaterte filer. For WampServer er dette *c:\wamp\www* og for MAMP er dette */Applications/MAMP/htdocs*. Filer som ikke er plassert her, kan ikke åpnes gjennom webserveren. Grunnen er enkel. Tenk deg at dette var en offentlig webserver. Du ville ikke at andre skulle fått tilgang til alle file på maskinen.
- Du finner frem til nettsidene ved å angi nettadresser der *http://localhost* tilsvarer *c:/wamp/www*. Ettersom fila i vårt eksempel heter *datotest.php* og ligger i en undermappe kalt *kapittel10*, får vi nettadressen *http://localhost/kapittel10/datotest.php*. For MAMP-brukere tilsvarer *http://localhost:8888* mappa */Applications/MAMP/htdocs*.
- Du kan ikke åpne fila i nettleseren ved å dobbeklikke på den slik vi gjorde med statiske filer. Gjør du det vil ikke fila åpnes gjennom webserveren, og koden blir ikke utført. Du vil med andre ord se resultatene av HTML-kodene, men ikke resultatet av PHP-kodene dersom du åpner den som en fil.



TIPS

Du kan enkelt se om en dynamisk nettside er åpnet på riktig måte ved å kontrollere at nettadressen starter med *http://*. Starter den med *file://* er den åpnet gjennom filtilgang.

Mer om visning av dato og tid

La oss nå utvide eksempelet noe, og legge til klokkeslett også. Den lille PHP-koden vi tidligere skrev inneholder et mønster for hvordan dato skal presenteres:

"**d/m/y**"

Her tilsvarer **d** dagen, **m** måneden og **y** året. Mellom disse har vi valgt å sette en /

for å få en fin formatering. Timer indikeres med en *H* (stor *H* for 24-timers format, og liten *h* for 12-timers format), minutter indikeres med en *i* (*m* er alt "opptatt" for måneder) og sekunder med en *s*. Vi kan dermed lage et mønster for presentasjon som ser slik ut:

"d/m/y H:i:s"

1. Fortsett redigeringen av fila *datotest.php* i din kodeeditor
2. Endre mønsteret for presentasjonen:

```
<p>Dagens dato er: <?php echo date("d/m/y H:i:s") ; ?></p>
```

3. Lagre fila, og test nettsiden på nytt i nettleseren. Sjekk at klokkeslett også vises:



Som du sikkert legger merke til så går ikke klokka. Dette er fordi koden kun utføres hver gang nettsiden genereres av webserveren. Trykker du oppdater i nettleseren vil nettsiden hentes på nytt og klokkeslettet oppdateres.

Ønsker du en klokke som går kontinuerlig må vi heller benytte *JavaScript*, som utføres i nettleseren og er et *client-side script*. Dette er ikke en del av denne boka. Som vi skal se er imidlertid PHP svært egnet til å utføre engangshendelser, slik som å hente ut data fra en database.

Skulle klokka gå feil har dette med tidssoneinnstillingen i webserveren å gjøre. Åpner du fila *C:\wamp\bin\apache\apacheX.X.X\bin\php.ini* kan du endre linja *date.timezone* til *date.timezone = Europe/Oslo*. Lagre og starte WampServer på nytt.

TIPS

Lære mer om PHP-koder

Selv om vi kommer til å bruke en del kodebiter av PHP er det ikke denne bokas mål at du skal lære deg å programmere i PHP. Vi kommer i stedet til å gi deg en del maler på kode som du kan benytte.

Dersom du synes PHP-koding virker interessant og ønsker å lære mer om å skrive slik kode ut over det som introduseres i denne boka, finnes det mange nettsteder om temaet. Et godt sted å starte er W3Schools sine PHP-tutorials:

<http://www.w3schools.com/php/>

11 Arbeide med MySQL

I dette kapitlet vil du lære

- hvordan du installerer MySQL Workbench
- å opprette prosjekter
- å lage tabeller
- hvordan du ser på innholdet i tabeller
- hvordan du modifiserer innholdet i tabeller
- om alternativet phpMyAdmin

Vi skal i denne boka først og fremst benytte dynamiske nettsider til å hente ut og legge inn data i en database. Vi vil forklare grundig hvordan dette gjøres, men selve teorien om hva en database er, og de ulike databasebegrepene, vil ikke tas like grundig. Dette ansees som en forkunnskap for innholdet i denne boka. Du kan finne både en introduksjon og en mer utdypende forklaring om databaser i f.eks. boka *IT-1, basisbok*.

Brukergrensesnitt til MySQL

Som tidligere nevnt er MySQL en tjeneste på maskinen på lik linje med Apache. Dette medfører at produktet har et begrenset grafisk brukergrensesnitt, da det er ment som tjenestetilbyder for andre programmer.

Apache har sitt grafiske brukergrensesnitt gjennom nettleseren, der vi kan se og navigere mellom nettsider Apache presenterer. På samme måte trenger vi et grafisk brukergrensesnitt til MySQL.

Her finnes det en mengde ulike alternativer – alt fra kommandolinjeverktøy der vi kun benytter tekstlige kommandoer, til webbaserte løsninger. WampServer tilbyr begge disse ytterpunktene gjennom *MySQL console* og *phpMyAdmin*. I dette kapitlet skal vi imidlertid presentere en applikasjon som kalles *MySQL Workbench*, for å få et litt mer tradisjonelt brukergrensesnitt i form av en applikasjon.

MySQL Workbench er ikke en del av WampServer, så denne må lastes ned separat. Verktøyet er fritt tilgjengelig og kan lastes ned fra MySQL-organisasjonens hjemmesider. Da denne boka ble skrevet, kunne man finne nedlastingene av *Workbench Community Edition* på denne nettadressen:

<http://dev.mysql.com/downloads/workbench/>

Verktøyet finnes både for Windows, Linux og Mac. Du kan bli spurta om å registrere en konto, men MySQL har også et valg som heter *No thanks, just take me to the downloads!*, dersom du ikke ønsker å gjøre dette.

Installasjonen av Workbench skulle være ganske rett fram og trenger ingen grundigere forklaring. Pass imidlertid på at du velger *Complete installation*. På det tidspunktet da denne boka ble skrevet, var 6.3.7 den nyeste versjonen av Workbench. Alle skjermbilder og framgangsmåter i denne boka vil derfor være fra denne versjonen.

Dersom du får advarsel om at *msvcr120.dll* mangler, må du installere *Visual C++ Redistributable Packages* for Visual Studio 2013. Installer både versjonen for 32-bit og 64-bit dersom du har et 64-bit operativsystem. Du finner produktene her: <https://www.microsoft.com/en-GB/download/details.aspx?id=40784>

TIPS

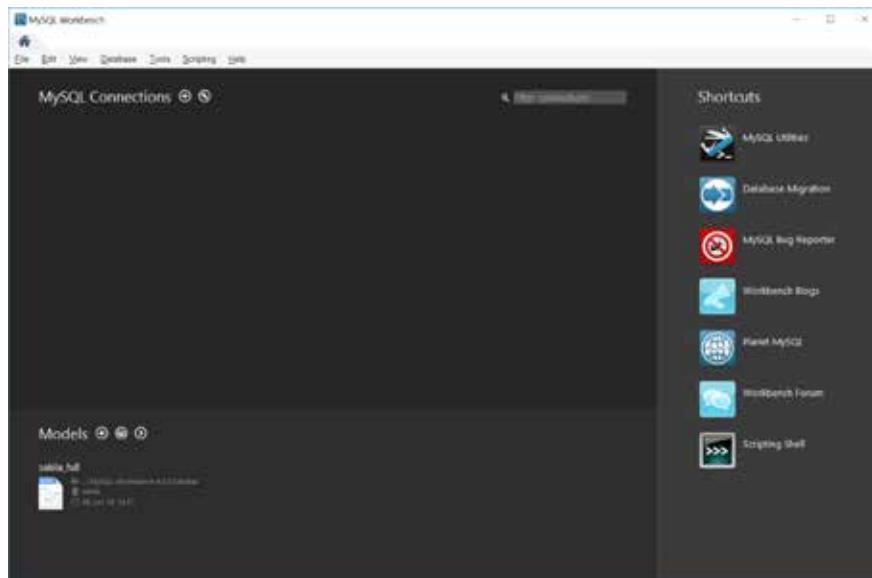
Du kan også være nødt til å installere *Microsoft Visual C++ 2010 Redistributable Package* og *.Net Framework 4.0*. Dette finner du her:

<https://www.microsoft.com/en-us/download/details.aspx?id=5555>

<https://www.microsoft.com/nb-no/download/details.aspx?id=17718>

TIPS

Dersom du starter Workbench fra startmenyen, får du følgende brukergrensesnitt:



Dette brukergrensesnittet består egentlig av flere forskjellige verktøy. Vi kan her utføre spørninger og opprette databasetabeller, modellere databaser samt administrere databaseserveren. I dette kapitlet skal vi holde oss til å behandle data, samt enkel administrasjon av databasen. Til slutt i denne delen av boka er det et kapittel om datamodellering med Workbench.

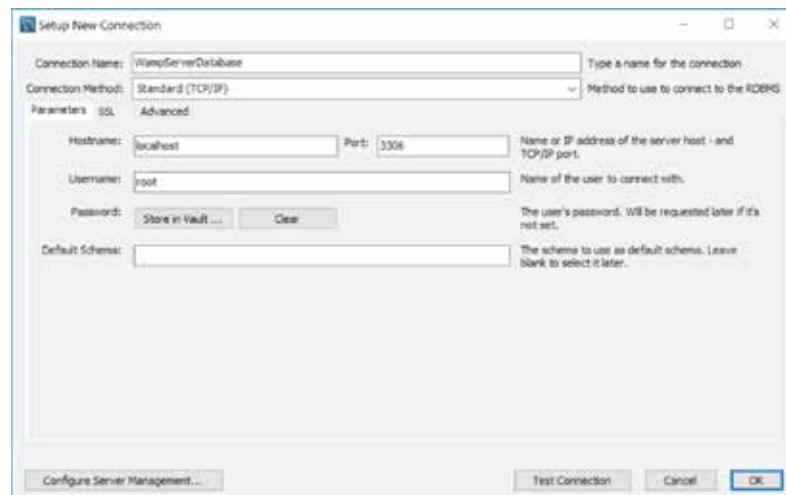
Før vi kan begynne å jobbe mot databasen, må vi imidlertid koble grensesnittet Workbench mot selve MySQL-databasen i WampServer. Dette er en engangsoperasjon der du legger inn tilkoblingsinformasjonen. Deretter kan du enkelt starte tilkoblingen når denne informasjonen er på plass.



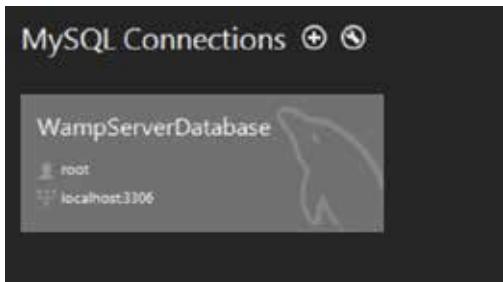
Opprette tilkobling til MySQL i Workbench

Som vi har nevnt, kan vi i Workbench lagre tilkoblingsinformasjonen for en database. Dette er en engangsoperasjon for hver database vi ønsker å koble til.

1. Sørg for at WampServer er startet
2. Klikk på knappen i seksjonen **MySQL Connections** MySQL Connections
3. Vi må nå velge et navn vi vil lagre tilkoblingsinformasjonen under, noe som til en viss grad kan sammenlignes med tittelen på et bokmerke for en nettadresse. Dette navnet velger vi selv, men benytt *WampServerDatabase* for å følge fremgangsmåten i denne boka. Benytt så *localhost* som **Hostname**, ettersom databasen ligger på samme maskin som Workbench kjører på. Som **Username** setter vi *root*, da dette er en standard administrasjonskonto som allerede finnes i databasen. Kontoen *root* har ikke noe passord som standard i WampServer



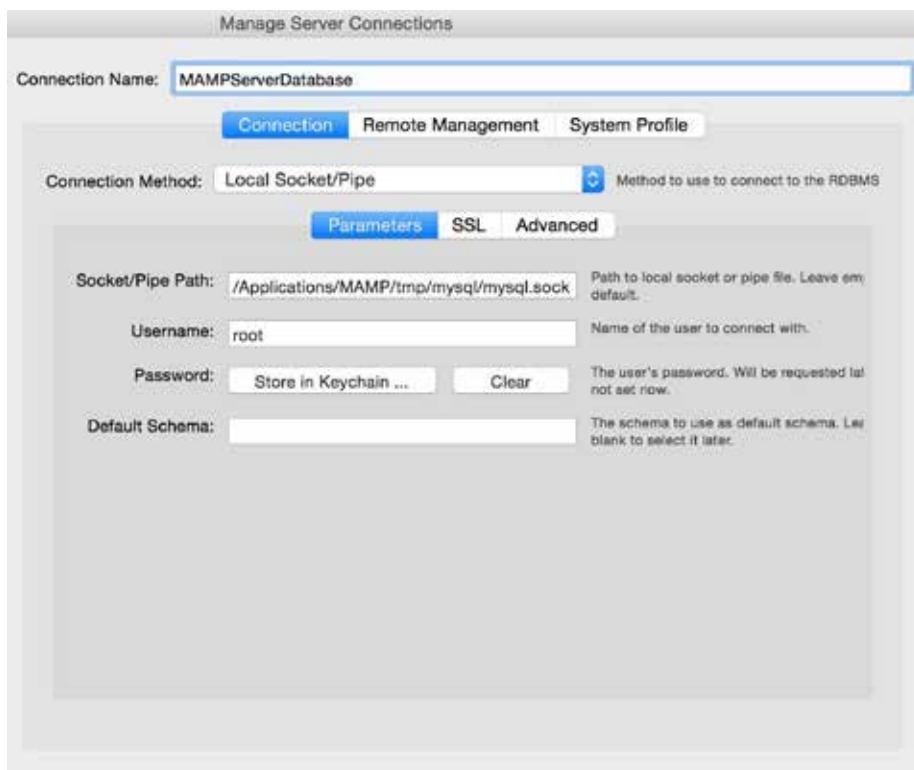
4. Klikk så på **Test Connection** for å sjekke at Workbench klarer å koble seg til databasen med informasjonen du har angitt, og velg så å lagre tilkoblingsinformasjonen ved å trykke **OK**
5. Du vil nå få inn tilkoblingen i listen over tilkoblinger:



Opprette tilkobling til MySQL i MAMP

Vær klar over at i Workbench sammen med MAMP kan det være at du må benytte sockets istedenfor portnummer. Bytt i så tilfelle tilkoblingstype fra *Standard(TCP/IP)* til *Local Socket/Pipe*, og velg socketen som vanligvis er lokalisert her:

`/Applications/MAMP/tmp/mysql/mysql.sock`

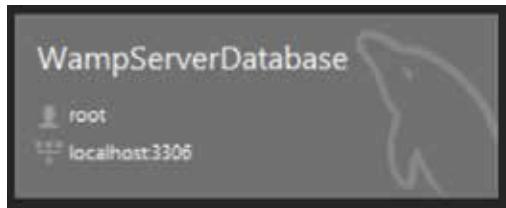


På MAMP vil også brukeren *root* ofte ha satt et passord som også er *root*. Dette kan du angi ved å trykke **Store in Keychain...**

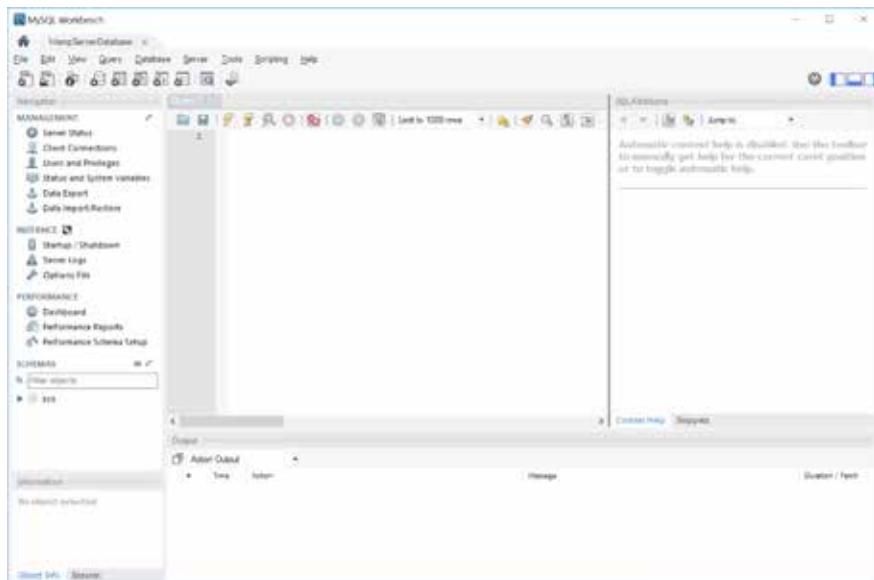
Koble til MySQL fra Workbench

Hver gang vi nå skal jobbe med databasen, må vi starte tilkoblingen som vi har laget.

- Dobbeltklikk på tilkoblingen *WampServerDatabase* på startskjermen



- Brukergrensesnittet åpner nå en ny fane der vi kan arbeide mot databasen



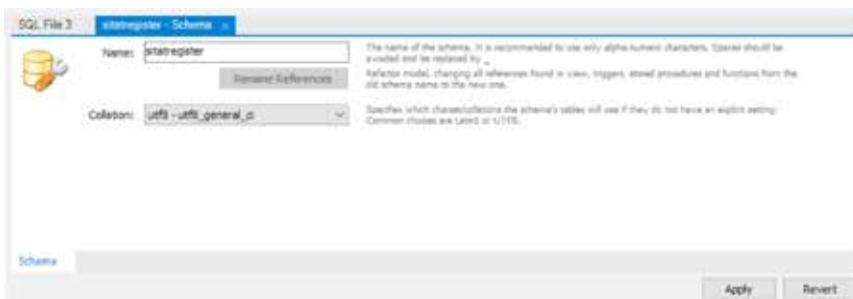
Opprette et nytt prosjekt

Før vi går i gang med å lage databasetabeller, må vi opprette et *schema* som samler sammen tabellene vi lager til et prosjekt. I den første delen av denne boka skal vi arbeide med en database som inneholder sitater, og vi velger derfor å gi prosjektet navnet *sitatregister*. For hvert nye prosjekt du lager, bør du opprette et slikt schema.

- Sørg for at du er logget inn i MySQL via Workbench
- Trykk på knappen for å lage et nytt schema (prosjekt) på verktøylinjen



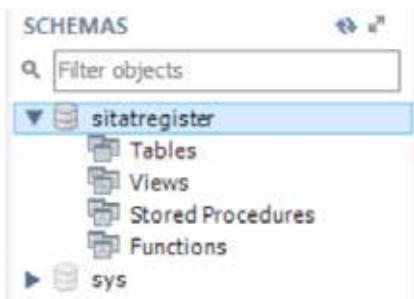
3. Skriv inn *sitatregister* som navn på det nye prosjektet. Velg også *utf8 - utf8_general_ci* som **Collation** for å være sikker på at vi benytter UTF8 som tegnsett. Klikk på **Apply**



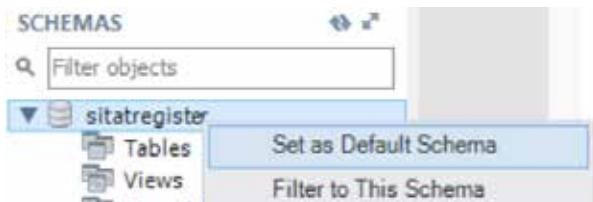
4. Klikk så på **Apply**, og deretter **Finish** i neste dialogboks som ber deg bekrefte operasjonen som skal utføres i databasen. Lukk deretter fanen for nytt schema (fanen har nå fått navnet *sitatregister - Schema*)
5. Du har nå fått et nytt prosjekt (schema) inn i MySQL-databasen, noe du kan se i panelet **SCHEMAS** (prosjektpanelet)

Et skjema omtales også av mange som en *database*. Dette er noe forvirrende da man også omtaler systemet MySQL som en *database*. I så fall må man kalle MySQL for et *DBMS (Database Management System)*. I denne boka kommer vi derfor til å benytte *prosjekt* som det norske navnet på *schema*.

TIPS



6. Høyreklikk på *sitatregister* i panelet **SCHEMAS**, og velg **Set as Default Schema** for å angi at videre handlinger skal gjelde dette prosjektet



Opprette databasetabeller

Før vi kan begynne å fylle databasen med data, må vi opprette en eller flere databasetabeller som vi kan legge dataene inn i. Vi skal nå lage en tabell som inneholder sitater, med *id* som primærnøkkel (identifikator) og selve sitatet som verdi. Det er vanlig å navngi tabeller basert på hva én rad i tabellen inneholder, så derfor velger vi navnet *sitat*.

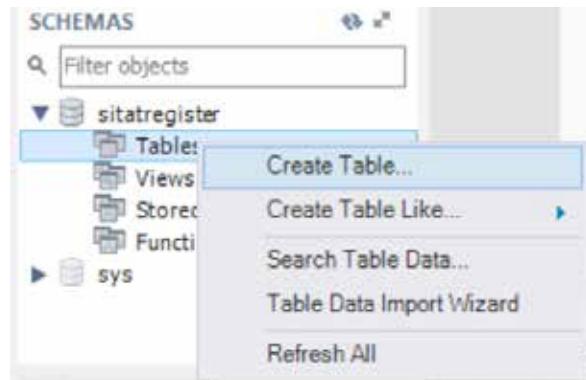
Et utdrag av databasetabellen *sitat* kan se slik ut:

id	tekst
1	Believe those who are seeking the truth; doubt those who find it
2	Just because something doesn't do what you planned it to do doesn't mean it's useless
3	I have not failed. I've just found 10,000 ways that won't work
...	...

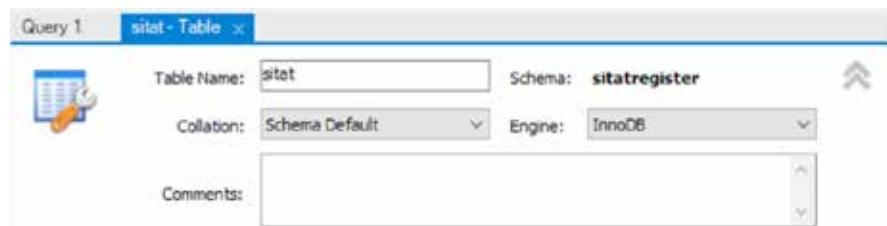
Rent formelt kan vi ved hjelp av EER-modellering presentere tabellen slik:



- Høyreklikk på **Tables** under prosjektet *sitatregister*, og velg **Create Table...**



- Gi tabellen navnet *sitat*



- Sørg for at fanen **Columns** er valgt nederst i dialogboksen, og skriv inn *id* som **Column Name** for den første kolonnen i tabellen *sitat*

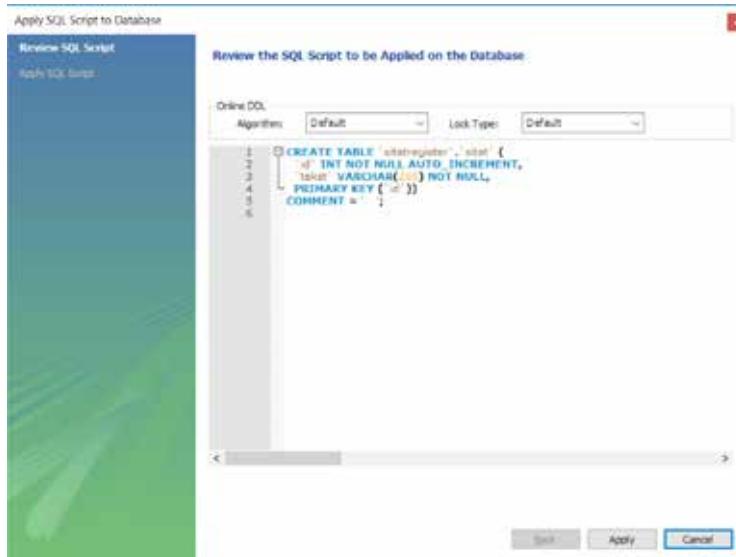
4. Workbench vil gjøre et forsøk på å fylle ut resten av innstillingene for deg. Ettersom dette er eneste kolonne, vil den settes til *primærnøkkel* (PK) og *not null* (NN). Workbench gjetter også på at datatypen skal være *INT*(heltall). Huk også av for at denne kolonnen skal være av typen *autoincrement* (AI), slik at oppføringene nummereres automatisk

Column Name:	Datatype:	PK	NH	UQ	B	UN	ZF	AI	G	Default/Expression:
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					

5. Legg til enda en kolonne med navnet *tekst*. La datatypen her være *VARCHAR(255)* slik at vi kan lagre opptil 255 tegn. Sørg også for at kolonnen er satt til *not null* for å sikre at det blir angitt en tekst på hvert sitat

Column Name:	Datatype:	PK	NH	UQ	B	UN	ZF	AI	G	Default/Expression:
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
tekst	VARCHAR(255)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					

6. Trykk så på **Apply**, ettersom vi nå er ferdige med tabelldefinisjonen
 7. Workbench vil nå vise deg kommandoene den vil kjøre mot MySQL-databasen. Godta disse ved å trykke på **Apply**, slik at tabellen blir overført til databasen. Avslutt så ved å trykke på **Finish**

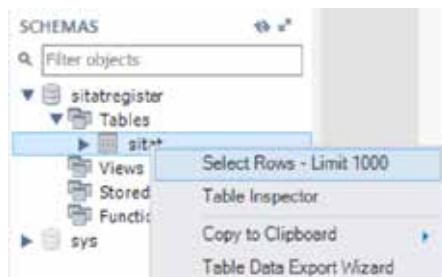


8. Lukk så fanen som nå heter *sitat*
 9. Som du ser i prosjektpanelet, har vi nå fått inn tabellen i databasen

Legge inn data i databasetabeller

Vanligvis ville vi laget en nettside der behandlingen av data kunne foregå. Det er allikevel kjekt å fylle databasen med litt testdata før vi kommer så langt. Vi skal derfor benytte Workbench til å legge inn de tre sitatene fra eksempladataene.

1. Høyreklikk på tabellen *sitat* under prosjektet *sitatregister*, og velg **Select Rows - Limit 1000**



2. Fyll ut tabellen med eksempladataene for de tre radene:

The screenshot shows the MySQL Workbench 'Result Grid' editor for the 'sitat' table. The table has two columns: 'id' and 'tekst'. There are three rows of data:

- Row 1: id=1, tekst='Believe those who are seeking the truth; doubt those who find it'
- Row 2: id=2, tekst='Just because something doesn't do what you planned it to do doesn't mean i...'
- Row 3: id=2, tekst='I have not failed. I've just found 10,000 ways that won't work.'

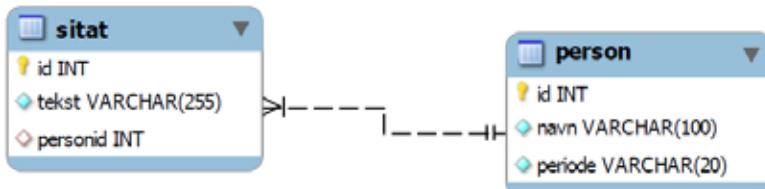
The editor includes a toolbar at the top with buttons for 'Edit', 'Export/Import', and 'Wrap Cell Contents'. On the right side, there's a vertical panel with tabs for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, there are 'Apply' and 'Revert' buttons.

3. Radene er ennå ikke overført til MySQL-databasen og eksisterer kun i applikasjonen. Trykk derfor på knappen **Apply** for å overføre dataene
4. Aksepter koden som skal utføres, ved å trykke på **Apply** i dialogboksen som nå kommer opp, og lukk dialogboksen ved å trykke på **Finish**
5. De tre radene er nå overført til databasen. Du kan nå lukke fanen for dataeditering av *sitat*-tabellen

Flere databasetabeller

Etter en stund kommer vi nok fram til at det ville være praktisk også å ha med hvem sitatene kommer fra i databasen. Dette kunne gjøres ved å legge til en ekstra kolonne i tabellen *sitat* for navn, men en bedre løsning vil være å ha en egen tabell for personer. Deretter kan vi koble disse to tabellene sammen gjennom en relasjon.

Tegnet med EER-modellering vil vår nye databasestruktur se slik ut:



Tabellen *person* kan se slik ut fylt med data:

id	navn	periode
1	Andre Gide	1869 - 1951
2	Thomas A. Edison	1847 - 1931
...

Som du ser, registrerer vi *id*, *navn* og *periode* for hver person.

- Høyreklikk på **Tables** under prosjektet *sitatregister*, og velg **Create Table...**
- Skriv inn navnet *person* på tabellen, samt følgende tabellstruktur:

Column Name	Datatype	PK	NN	UQ	B	UN	ZP	AE	G	Default/Expression
<i>id</i>	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<i>navn</i>	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<i>periode</i>	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

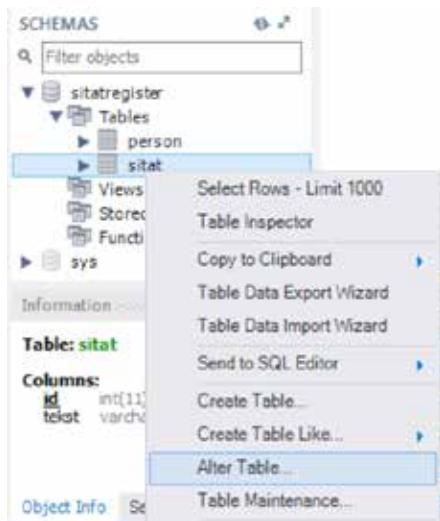
- Velg **Apply**, og godkjenn utførelsen gjennom å velge **Apply** og **Finish**. Lukk så fanen
- Høyreklikk på tabellen *person*, og velg **Select Rows - Limit 1000**
- Legg så inn følgende eksempladata i tabellen *person*:

Result Grid			
	id	navn	periode
	1	Andre Gide	1869 - 1951
	2	Thomas A. Edison	1847 - 1931
		HULL	HULL

- Velg til slutt knappen **Apply** for å overføre dataene til MySQL, og bekrefte med **Apply** og **Finish**. Lukk til slutt editeringsfanen for *person*-tabellen

Vi må nå også gjøre en endring på *sitat*-tabellen for å kunne koble sitatene og personer. Vi ønsker med andre ord å lage en fremmednøkkel fra en ny kolonne i *sitat*-tabellen som vi kaller *personid* til *id*-kolonnen i *person*-tabellen. Dette vil gjøre at vi kun får lov til å oppgi en *personid* i *sitat*-tabellen dersom personen eksisterer i *person*-tabellen.

1. Høyreklikk på tabellen *sitat*, og velg **Alter Table...** for å endre strukturen på denne tabellen



2. Legg til en tredje kolonne som heter *personid*. Denne kolonnen må tillate *NULL*-verdier, ettersom vi har registrert sitater som ennå ikke har noen person knyttet til seg

Column Name	Datatype	PK	NH	UQ	B	UN	ZF	AI	G	Default/Expression
<i>id</i>	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
tekst	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
<i>personid</i>	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

3. Overfør endringene til databasen ved å trykke på **Apply**, **Apply** og **Finish**, men lukk ikke dialogboksen
4. For å få en relasjon mellom *personid* i tabellen *sitat* og primærnøkkelen *id* i tabellen *person* velger du fanen **Foreign Keys** i dialogboksen

I enkelte versjoner av Workbench/MySQL blir ikke alltid tabellmotoren satt til *InnoDB*, selv om dette stod oppført som standard under tabellopprettelsen. Får du en melding om at tabellen ikke støtter fremmednøkler, så bytt til *InnoDB* under **Engine** og trykk **Apply**, **Apply**, og **Finish**. Dette må gjøres på alle tabeller som skal kobles.



5. Skriv inn *FK_sitat_person* som navn på fremmednøkkelen og velg *sitatregister.person* som tabellen det skal refereres til



6. Huk så av for at det er *personid* som er kolonnen som skal referere til noe, og velg så at denne da refererer til *id*-kolonnen i *person*-tabellen



7. Velg så **Apply**, **Apply** og **Finish**. Lukk deretter dialogboksen
8. Høyreklikk på *sitat*-tabellen i prosjektpanelet, og velg **Select Rows - Limit 1000**
9. Som du ser, har vi nå fått inn en ny kolonne i tabellen, der verdiene foreløpig er satt til *NULL* (databasebegrepet for «ingen verdi»)

Result Grid			
	id	tekst	personid
▶	1	Believe those who are seeking the truth; doubt those who find it	NULL
	2	Just because something doesn't do what you planned it to do doesn't mean it's useless	NULL
	3	I have not failed. I've just found 10,000 ways that won't work	NULL
*	NULL	NULL	NULL

10. Fyll ut kolonnen *personid* med hvilken person sitatet tilhører

Result Grid			
	id	tekst	personid
▶	1	Believe those who are seeking the truth; doubt those who find it	1
	2	Just because something doesn't do what you planned it to do doesn't mean it's useless	2
	3	I have not failed. I've just found 10,000 ways that won't work	2
*	NULL	NULL	NULL

11. Send endringene over til databasen ved å trykke på **Apply**, **Apply** og **Finish**
12. For å sikre at et sitat i framtiden ikke kan legges inn uten å være knyttet til en person, høyreklikker vi på tabellen *sitat* og velger **Alter Table...** Huk så av for at vi ikke tillater *NULL*-verdier for *personid*-feltet

Column Name	Datatype	PK	NN	UQ	B	UN	ZP	AE	G	Default/Expression:
<i>id</i>	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<i>tekst</i>	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<i>personid</i>	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

13. Send endringene til databasen ved å klikke på **Apply**, **Apply** og **Finish**
14. Lukk til slutt fanen

Slette rader

Dersom du ønsker å slette en rad fra databasen, kan også dette gjøres fra Workbench. Vi skal her slette et ekstra sitat vi har lagt til for *Thomas Edison*. Pass på at du etter denne øvingen står igjen med de tre opprinnelige sitatene.

1. Høyreklikk på tabellen *sitat*, og velg **Select Rows - Limit 1000**, slik du ville ha gjort for å legge til nye rader eller endre data
2. Høyreklikk på raden/radene du ønsker å slette, og velg **Delete Row(s)** på menyen

The screenshot shows a MySQL Workbench interface with a table named 'sitat' containing four rows. The fourth row, which has the value 'The chief function of the laws of gravity is not to protect the stupid from falling, but to enable the stupid to fall without breaking their necks.' in the 'tekst' column, is selected. A context menu is open over this row, with the 'Delete Row(s)' option highlighted in blue. Other options in the menu include 'Open Value in Editor', 'Set Field to NULL', 'Mark Field Value as a Function/Literal', 'Load Value From File...', and 'Save Value To File...'. The top of the window shows tabs for 'Result Grid', 'Edit', 'Export/Import', and 'Wrap C'.

id	tekst	personid
1	Believe those who are seeking the truth; doubt those who find it.	1
2	Just because something doesn't do what you planned it to do doesn't mean it's useless.	2
3	I have not failed. I've just found 10,000 ways that won't work.	2
4	The chief function of the laws of gravity is not to protect the stupid from falling, but to enable the stupid to fall without breaking their necks.	2

3. Selv om raden nå ser fjernet ut, må du trykke på **Apply**, **Apply** og **Finish** for å overføre endringene (slettingene) til databasen

Opprette brukere

Brukeren *root* som vi til nå har benyttet for å koble til MySQL, har et par farer ved seg. Dette er en superbruker som har alle rettigheter i databasen. Derfor er det ofte hensiktsmessig å opprette en ny bruker for hvert prosjekt, som nettsidene vi senere skal lage, benytter.

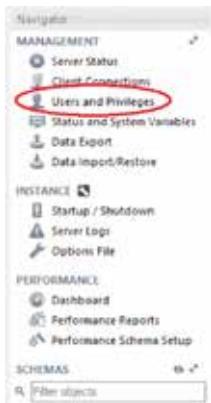
Denne nye brukeren gis da begrensede rettigheter som kun samsvarer med de operasjonene vi ønsker utføre fra nettsiden. Vi skal fortsatt benytte brukeren *root* for å administrere MySQL gjennom Workbench, men om noen skulle klare å få tak i kodene bak websidene vi lager, og dermed også brukernavnet og passordet som benyttes, kan de ikke gjøre så stor skade som de ville gjort med root-brukeren.

Vi skal derfor opprette en ny bruker med brukernavnet *webgrensesnitt*. Denne brukeren skal kun få lov til å hente ut (*SELECT*), sette inn (*INSERT*), endre (*UPDATE*) og slette (*DELETE*) data i tabellene som omfattes av prosjektet *sitatregister*. Disse innstillingene passer godt med det vi skal lage av funksjonalitet i nettsidene i de neste kapitlene. Brukeren skal også være passordbeskyttet med passordet *drossap*.

For enkelhets skyld kommer resten av boka til å benytte brukeren *root* uten noe passord i eksemplene. Du kan imidlertid gjerne benytte brukeren *webgrensesnitt* med passordet *drossap* isteden, for prosjekter mot prosjektet *sitatregister*. Tilsvarende kan du lage nye brukere for de andre prosjektene.

Legge til brukere og rettigheter

- Velg **Users and Privileges** i venstremenyen

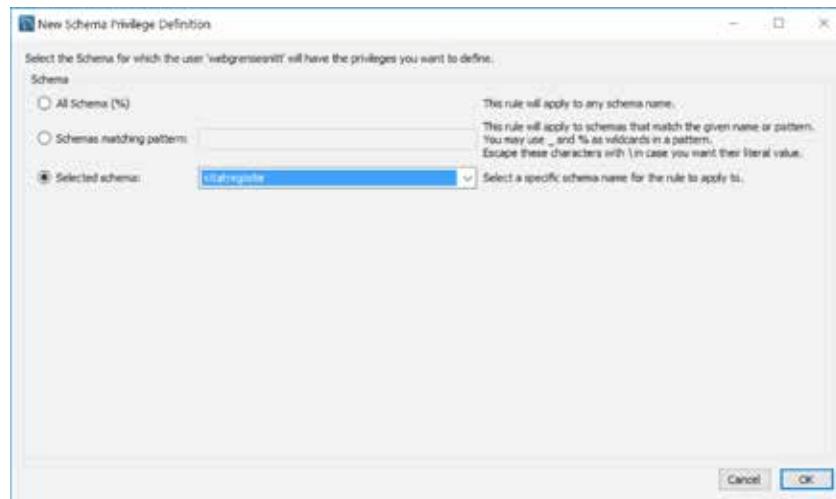


- Trykk på **Add Account** helt nederst til venstre under listen over eksisterende kontoer, og fyll ut med *webgrensesnitt* som brukernavn og *drossap* som passord. Sett også **Limit to Hosts Matching** til *localhost*

User	From Host
root	localhost
newuser	%

- Trykk så på **Apply** for å legge til brukeren
- Velg fanen **Schema Privileges**

5. Trykk så på knappen **Add Entry...** for å legge til en ny rettighet for brukeren **webgrensesnitt**, velg **sitatregister** som **Selected schema**, og bekrefte med **OK**



6. Mens denne nye regelen er markert, huk av for å gi brukeren **SELECT, INSERT, UPDATE** og **DELETE**-rettigheter



7. Oversend informasjonen om rettigheter til MySQL ved å trykke på **Apply**

phpMyAdmin

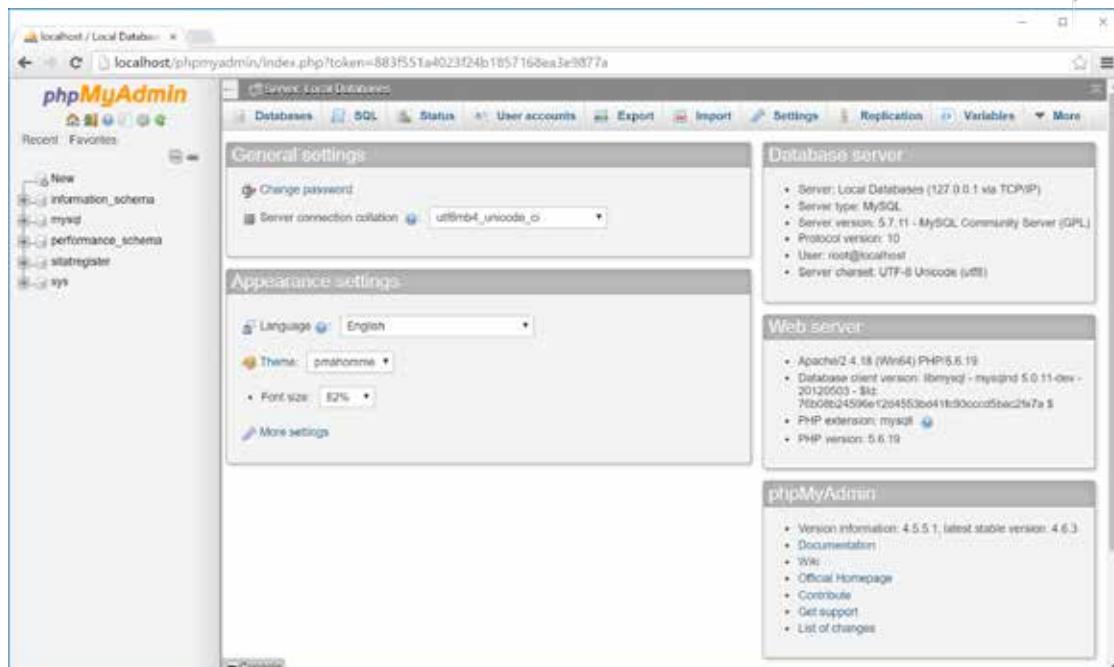
Et alternativ til å benytte Workbench er å benytte et verktøy kalt *phpMyAdmin*. Dette er et webbasert administrasjonspanel/brukergrensesnitt for en MySQL-database, som kan startes fra menyen til WampServer. Du vil i phpMyAdmin finne igjen alle funksjonene som er beskrevet i dette kapitlet. Du logger inn med brukernavnet *root* og blankt passord.

I MAMP finner du en link til phpMyAdmin på startsiden:

<http://localhost:8888/MAMP/?language=English>

Husk at brukeren *root* også har passordet *root* på enkelte varianter av MAMP

TIPS



Resten av denne boka vil benytte MySQL Workbench i eksemplene. Det er imidlertid ingen ting i veien for å gjøre de samme operasjonene gjennom phpMyAdmin, da begge disse kun er grensesnitt mot databasen.

Du finner mer informasjon om phpMyAdmin på deres nettsider:

<http://www.phpmyadmin.net/>

TIPS

Noen tjenestetilbydere for nettsider/webserverplass vil ikke tilby tilkobling fra MySQL Workbench til sin MySQL-database. De vil isteden tilby administrasjon av databasen via et phpMyAdmin-grensesnitt. Dette grensesnittet finner du på en nettadresse som de vil opplyse om.

Datatyper i MySQL

Som du sikkert har lagt merke til når vi har opprettet tabeller i Workbench, inneholder MySQL en rekke datatyper tilpasset ulike verdier. Når man velger datatype for en kolonne, er det viktig å velge en datatype som tar minst mulig lagringsplass, men samtidig tar vare på verdiene man ønsker.

Vi skal her gi en kort oversikt over de mest brukte datatypene:

- **BOOLEAN:** Holder på verdiene true/false. Nyttig til f.eks. å ta vare på informasjon slik som «ønsker å bli kontaktet pr. e-post»
- **INTEGER:** Heltall mellom -2147483648 og 2147483647
- **FLOAT:** Desimaltall (flyttall)
- **DATETIME:** Holder på dato og tid med formatet *YYYY-MM-DD HH:MM:SS*
- **CHAR(X):** Holder på X antall tegn. Dersom man ikke benytter hele antallet tegn i teksten, vil man få et tilsvarende antall blanke tegn (whitespaces) lagt til
- **VARCHAR(X):** Kan holde på et variabelt antall tegn, men ikke flere enn det som er angitt ved X. Den største verdien X kan ha, er i 5.7-versjonen av MySQL 65,535. Allikevel er det av vanlig å gå over til datatypen TEXT når X overskridet 255
- **TEXT:** Holder på en større tekst, samt spesialtegn som tabulator og linjeskift

Dette er kun et lite utvalg av datatypene, og de fleste datatypene kommer også i flere versjoner. Du finner mer informasjon om datatypene til MySQL i dokumentasjonen:

<http://dev.mysql.com/doc/refman/5.7/en/data-types.html>

12 Koble dynamiske nettsider og MySQL

I dette kapitlet vil du lære

- hvordan du kobler en dynamisk nettside og en MySQL-database
- å hente ut et enkelt datasett basert på SQL-spørninger
- å presentere et datasett i tabellform

Å jobbe mot en database innebærer at vi må skrive en noen PHP-koder. Disse kodene kan i første omgang virke forvirrende og uforståelige, men du vil etter hvert se at det er de samme blokkene med kode som gjentar seg hver gang vi skal gjøre de ønskede operasjonene.

Det er som tidligere nevnt ikke innenfor denne bokas rammer å lære deg programmering i PHP. Vi skal altså bare benytte ferdige kodeblokker, som du også kan ta med deg videre i egne prosjekter. Bakerst i boka finnes det en oversikt over disse kodeblokkene. Også på nettsidene til boka vil du finne maler slik at du enklere kan klippe og lime. Malene på bokas nettsider lar deg også generere blokker med kode basert på dine egne opplysninger.

Når vi skal skrive inn koden, er det viktig å være svært nøyne med at det blir korrekt. Merk deg spesielt at koden er såkalt *case sensitiv* (dvs. skiller på små/store bokstaver). Det kan være lurt å teste nettsiden etter hver gang du har skrevet inn en ny blokk med kode, for å sjekke at koden ikke medfører noen feilmeldinger. Det er vanskeligere å finne feil dersom du venter til eksemplene er helt ferdige.

For å gjøre koden og eksemplene enkle for deg som nybegynner vil vi ikke ha fokus på sikkerhet og pålitelighet. Vi har også begrenset bruken av design og CSS for å gjøre prosjektene enklere å følge.



Koble til en database

Vi skal nå starte på et nytt prosjekt som vi også skal videreutvikle i de påfølgende kapitlene. Her skal vi koble oss til *sitatregister*-databasen vi laget i forrige kapittel. Vi skal i dette første kapittelet hente ut informasjonen om sitatene i tabellform på en nettside.

1. Åpne mal-filen du tidligere har laget i din kodeeditor
2. Legg til følgende koder som utfører en kobling til databasen først i dokumentet. Her oppgir vi plassering, brukernavn, passord og databasenavn.

```
<?php
    $stilkobling = mysqli_
connect("localhost","root","","sitatregister");
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Sidetittel</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <p>Innhold</p>
    </body>
</html>
```



Dersom du benytter MAMP kan det være at du må oppgi «root» også som passord der det i koden over står et blankt felt:

```
$stilkobling = mysqli_
connect("localhost","root","","root","sitatregister");
```

3. Lag en ny mappe i *c:\wamp\www* med navn *kapittel12*
4. Lagre koden som *sitaterenkel.php* i mappa *kapittel12*
5. Velg å åpne nettsiden i nettleseren ved å skrive inn nettadressen <http://localhost/kapittel12/sitaterenkel.php>. Nettsiden skal nå kun vise innholdet i **<body>**-taggen på vanlig måte



Dersom du benytter MAMP må du benytte nettadressen <http://localhost:8888/kapittel12/sitaterenkel.php>

6. Dersom det dukker opp en feilmelding kan dette både være fordi du har skrevet noe feil i koden eller fordi webserveren og databasen ikke er satt opp riktig

 Parse error: syntax error, unexpected '"root"' (T_CONSTANT_ENCAPSED_STRING) in C:\wamp\www\kapittel2\sitaterenkel.php on line 2

Hente ut en tabelloversikt av dataene

Koblingen mot MySQL i seg selv er ikke spesielt interessant. Det er først når vi begynner å benytte denne koblingen til å hente ut eller endre data, at vi får noen effekt av den. Derfor skal vi nå hente ut en tabelloversikt av sitatene fra databasen.

Tabelloversikt er en av mange mulige presentasjonsmåter. Vi skal ta for oss flere presentasjonsmåter i neste kapittel.



Framgangsmåten for å presentere data i tabellform består i først å lage et *datasett* med dataene vi ønsker å hente ut, og så knytte en tabell-presentasjon mot dette datasettet.

1. Legg inn følgene PHP-koder som henter ut datasettet i PHP-blokka vi alt har laget:

```
<?php  
    $tilkobling = mysqli_connect("localhost", "root", "", "sitatregister");  
    $sql = "SELECT id, tekst FROM sitat";  
    $datasett = $tilkobling->query($sql);  
?  
<!DOCTYPE html>  
<html>
```

2. Test nettsiden igjen, og kontroller at du får en tom nettside uten noen feilmeldinger
3. Legg inn en tabell i nettsiden med to rader og to kolonner. Første rad skal inneholde overskriftene *ID* og *Tekst*

```
<body>  
    <table>  
        <tr>  
            <th>ID</th>  
            <th>Tekst</th>  
        </tr>  
        <tr>  
            <td></td>  
            <td></td>  
        </tr>  
    </table>  
</body>
```

4. Legg også inn en liten stil i <head>-delen av nettsiden som gjør at vi ser kantlinjer på tabellen

```
<head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <style>
        table, th, td {
            border: 1px solid black;
        }
    </style>
</head>
```

5. Test nettsiden, og se at du har en tabell

ID	Tekst
1	Believe those who are seeking the truth; doubt those who find it

6. Angi ved å legge til følgende PHP-kode at hele den andre raden skal gjenta seg like mange ganger som det er oppføringer i datasettet. For hver gang skal de to cellene fylles med innholdet i datafeltene *id* og *tekst* i datasettet:

```
<body>
    <table>
        <tr>
            <th>ID</th>
            <th>Tekst</th>
        </tr>
        <?php while($rad = mysqli_fetch_array($datasett)) { ?>
        <tr>
            <td><?php echo $rad["id"]; ?></td>
            <td><?php echo $rad["tekst"]; ?></td>
        </tr>
        <?php } ?>
    </table>
</body>
```

7. Test nettsiden i nettleseren din. Kontroller at sitatene presenteres i tabellform, og at alle tre sitatene er med

ID	Tekst
1	Believe those who are seeking the truth; doubt those who find it
2	Just because something doesn't do what you planned it to do doesn't mean it's useless
3	I have not failed. I've just found 10.000 ways that won't work

TIPS

Får du problemer med at norske tegn i informasjon fra databasen ikke vises korrekt på nettsiden, så se seksjon om dette i *kapittel 15*.

13 Hente ut data

I dette kapitlet vil du lære

- om ulike typer datapresentasjon
- å koble data fra flere tabeller
- å lage søkefelt
- om skjemaer og skjemaelementer

I forrige kapittel så vi på hvordan man kobler en nettside til en MySQL-database, samt hvordan man kunne hente ut en enkel tabellpresentasjon av dataene.

Vi skal i dette kapitlet fortsette med å hente ut data fra databasen, men vi skal samtidig se på alternative presentasjonsformer for dataene på nettsiden, samt hvordan vi kan prosessere dataene ved hjelp av mer avansert SQL.

Denne boka kommer ikke til å forklare SQL i spesiell detalj. Det finnes derimot et kapittel i boka *IT-1, basisbok* som tar for seg det å skrive grunnleggende SQL-spørninger.

Koble flere tabeller

I vår sitat-database er dataene vi ønsker å presentere skilt i to databasetabeller med navn *sitat* og *person*. Vi ønsker nå å lage et datasett der dataene fra disse to tabellene settes sammen. På den måten kan vi presentere både sitatet og navnet på personen som har skrevet dette.

1. Åpne mal-filen i din kodeeditor
2. Legg inn koblingen til databasen

```
<?php
    $stilkobling = mysqli_
connect("localhost","root","","sitatregister");
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Sitetittel</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <p>Innhold</p>
    </body>
</html>
```

3. Lag datasettet som plukker ut *id*, *tekst*, *navn* og *periode* fra tabellene *person* og *sitat*. Fortell at vi også kun er interessert i de koblingene mellom sitater og personer, der sitatets *personid* er den samme som personen sin *id*. Dersom vi har registrert en person uten noen sitater, vil ikke denne vises i dette utplukke

```
<?php
    $stilkobling = mysqli_connect("localhost", "root", "", "sitatregister");
    $sql = "SELECT sitat.id, sitat.tekst, person.navn, person.periode
            FROM person, sitat
            WHERE sitat.personid = person.id";
    $datasett = $stilkobling->query($sql);
?>
<!DOCTYPE html>
<html>
```

4. Lag en ny mappe i *c:\wamp\www* med navn *kapittel13*
 5. Lagre koden som *sitateravansert.php* i mappa *kapittel13*
 6. Test nettsiden i nettleseren ved å skrive inn nettadressen
http://localhost/kapittel13/sitateravansert.php, og kontroller at du får en blank nettside uten feilmeldinger
 7. Legg inn en ny tabell i *sitateravansert.php* som du gir to rader og fire kolonner. La første rad inneholde kolonneoverskriftene *ID*, *Tekst*, *Navn* og *Periode*

```
<body>
    <table>
        <tr>
            <th>ID</th>
            <th>Tekst</th>
            <th>Navn</th>
            <th>Periode</th>
        </tr>
        <tr>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
        </tr>
    </table>
</body>
```

8. Legg også inn en liten stil i *<head>*-delen av nettsiden som gjør at vi ser kantlinjer på tabellen

```
<head>
    <title>Sidetittel</title>
    <meta charset="utf-8" />
    <style>
        table, th, td {
            border: 1px solid black;
        }
    </style>
</head>
```

9. Test nettsiden, og se at du har en tabell

ID	Tekst	Navn	Periode
1	Believe those who are seeking the truth; doubt those who fint it	Andre Gide	1869 - 1951

10. Angi at andre rad i tabellen skal gjentas for hver rad/oppføring i datasettet, og at cellene skal fylles med verdiene fra datasettet

```
<table>
  <tr>
    <th>ID</th>
    <th>Tekst</th>
    <th>Navn</th>
    <th>Periode</th>
  </tr>
  <?php while($rad = mysqli_fetch_array($datasett)) { ?>
  <tr>
    <td><?php echo $rad["id"]; ?></td>
    <td><?php echo $rad["tekst"]; ?></td>
    <td><?php echo $rad["navn"]; ?></td>
    <td><?php echo $rad["periode"]; ?></td>
  </tr>
  <?php } ?>
</table>
```

11. Test nettsiden i nettleseren, og sjekk at du får ut dataene fra databasen

ID	Tekst	Navn	Periode
1	Believe those who are seeking the truth; doubt those who fint it	Andre Gide	1869 - 1951
2	Just because something doesn't do what you planned it to do doesn't mean it's useless	Thomas A. Edison	1847 - 1931
3	I have not failed. I've just found 10.000 ways that won't work	Thomas A. Edison	1847 - 1931

Presentere data ved hjelp av regioner

Foreløpig har vi kun presentert data fra databasen i tabellform. Ofte ønsker vi en annen og mer tilpasset presentasjonsform på nettsiden. Vi skal derfor lage en ny nettside som viser det samme datasettet, men isteden benytter en såkalt *repeterende region*.

- Åpne mal-filen i din kodeeditor
- Legg inn samme kobling mot databasen og samme datasett som i fila *sitateravansert.php*. Om du ikke ønsker skrive inn koden på nytt, kan du kopiere denne koden

```
<?php
  $stilkobling = mysqli_
connect("localhost","root","","sitatregister");
  $sql = "SELECT sitat.id, sitat.tekst, person.navn, person.periode
        FROM person, sitat
        WHERE sitat.personid = person.id";
  $datasett = $stilkobling->query($sql);
?>
<!DOCTYPE html>
<html>
```

3. Legg inn følgende tekst i nettsiden:

```
<body>
  <p>
    Sitatet:<br />
    Ble skrevet av:<br />
    Som levde i perioden:
  </p>
</body>
```

4. Fortell så at denne tekstblokken (paragrafen) skal repeteres for hver oppføring i datasettet, og at tilhørende datafelter fra datasettet skal settes inn

```
<body>
  <?php while($rad = mysqli_fetch_array($datasett)) { ?>
  <p>
    Sitatet: <?php echo $rad["tekst"]; ?><br />
    Ble skrevet av: <?php echo $rad["navn"]; ?><br />
    Som levde i perioden: <?php echo $rad["periode"]; ?>
  </p>
  <?php } ?>
</body>
```

5. Lagre fila som *sitaterregion.php* i mappa *kapittel13*

6. Test nettsiden i nettleseren, og kontroller resultatet

```
Sitatet: Believe those who are seeking the truth; doubt those who find it
Ble skrevet av: Andre Gide
Som levde i perioden: 1869 - 1951

Sitatet: Just because something doesn't do what you planned it to do doesn't mean it's useless
Ble skrevet av: Thomas A. Edison
Som levde i perioden: 1847 - 1931

Sitatet: I have not failed. I've just found 10.000 ways that won't work
Ble skrevet av: Thomas A. Edison
Som levde i perioden: 1847 - 1931
```

7. Formater så ledetekstene i fet skrift og det dynamiske datafeltet *tekst* i kursiv skrift ved å legge til ** og ** rundt PHP-blokkene. Legg også til hermete tegn (*"*) foran og bak

```
<body>
  <?php while($rad = mysqli_fetch_array($datasett)) { ?>
  <p>
    <strong>Sitatet:</strong> <em>&quot;<?php echo $rad["tekst"]; ?>&quot;</em><br />
    <strong>Ble skrevet av:</strong> <?php echo $rad["navn"]; ?><br />
    <strong>Som levde i perioden:</strong> <?php echo $rad["periode"]; ?>
  </p>
  <?php } ?>
</body>
```

8. Test nettsiden i nettleseren, og kontroller resultatet

Sitatet: "Believe those who are seeking the truth; don't those who find it"
Ble skrevet av: Andre Gide
Som levde i perioden: 1869 - 1951

Sitatet: "Just because something doesn't do what you planned it to do doesn't mean it's useless"
Ble skrevet av: Thomas A. Edison
Som levde i perioden: 1847 - 1931

Sitatet: "I have not failed. I've just found 10.000 ways that won't work"
Ble skrevet av: Thomas A. Edison
Som levde i perioden: 1847 - 1931

Tilpassede datapresentasjoner

Vi skal nå tilpasse dataene enda mer til vårt bruk ved å presentere kun ett tilfeldig sitat på en nettside. Dette gjør vi ved å utvide SQL-spørringen til datasettet med SQL-kode som forteller systemet at det skal sortere sitatene i tilfeldig rekkefølge, før deretter å kun velge ut første sitat.

1. Åpne mal-filen i din kodeeditor
2. Legg inn koblingen til databasen

```
<?php
    $stilkobling = mysqli_connect("localhost", "root", "", "sitatregister");
?>
<!DOCTYPE html>
<html>
```

3. Lag et datasett som plukker ut ett tilfeldig sitat

```
<?php
    $stilkobling = mysqli_connect("localhost", "root", "", "sitatregister");
    $sql = "SELECT sitat.tekst, person.navn, person.periode
        FROM person, sitat
        WHERE sitat.personid = person.id
        ORDER BY rand() LIMIT 1";
    $datasett = $stilkobling->query($sql);
?>
<!DOCTYPE html>
<html>
```

4. Du skal nå sørge for at det blir satt inn en overskrift som skal vise sitatet. Legg derfor inn en **<h1>**-tag og sørг for at teksten står i hermetegn og kursiv. Legg også til en paragraf som inneholder navnet og perioden formatert ved hjelp av bindestrek og parenteser. Angi at dette skal gjøres for den ene oppføringen i datasettet. Dette angis med **if** i stedet for **while** som vi har benyttet tidligere

```
<body>
    <?php if($rad = mysqli_fetch_array($databaset)) { ?>
        <h1><em><?php echo $rad["tekst"]; ?></em></h1>
        <p>- <?php echo $rad["navn"]; ?> (<?php echo $rad["periode"]; ?>)</p>
        <?php } ?>
</body>
```

5. Lagre nettsiden som *tilfeldigsitat.php* i mappa *kapittel13*
 6. Test nettsiden, og se til at du får ut et sitat. Trykk så på oppdater i nettleseren flere ganger, og kontroller at andre sistater velges tilfeldig

"Just because something doesn't do what you planned it to do doesn't mean it's useless"

- Thomas A. Edison (1847 - 1931)

Søk i data

Vi skal her lage en svært enkelt utgave av søk blant data. Nettsiden vil ved oppstart kun vise en søkerboks. Dersom vi skriver noe i søkerboksen og klikker på *Søk*, vil nettsiden vise sitater som inneholder angitt søkerstreng.

- Åpne mal-filen i din kodeeditor
- Legg inn koblingen til databasen

```
<?php
    $stilkobling = mysqli_
connect("localhost","root","","sitatregister");
?>
<!DOCTYPE html>
<html>
```

- Legg så inn et skjema i nettsiden. Dette skjemaet skal ha en tekstboks, og en knapp for å utføre søk med

```
<body>
    <form method="get">
        <label for="txtSøkestreng">Søkestreng:</label>
        <input type="text" name="txtSøkestreng" id="txtSøkestreng" />
        <button type="submit" name="submit">Søk</button>
    </form>
</body>
```

TIPS

En mer utdypende forklaring av HTML-skjemaer kommer til slutt i dette kapittelet.

4. Lagre fila som *soek.php* i mappa *kapittel13*
5. Test nettsiden i en nettleser, og kontroller at du får opp skjemaet

Søkestreng:

6. Legg til et datasett som kun plukkes ut dersom *submit*-knappen er trykket.

```
<?php
    $stilkobling = mysqli_connect("localhost", "root", "", "sitatregister");

    if(isset($_GET["submit"]))
    {
        $sql = sprintf("SELECT tekst
                        FROM sitat
                        WHERE tekst LIKE '%%%" . $_GET["txtSokestreng"] . "%'",
                        $stilkobling->real_escape_string($_GET["txtSokestreng"]));
    }
    $datasett = $stilkobling->query($sql);
}

?>
<!DOCTYPE html>
<html>
```

Ettersom uttrykket **LIKE** i SQL benytter %-tegn slik som også formateringsfunksjonen i PHP gjør, må vi benytte doble %-tegn i SQL-koden vi skriver i PHP (såkalt *escape character*)

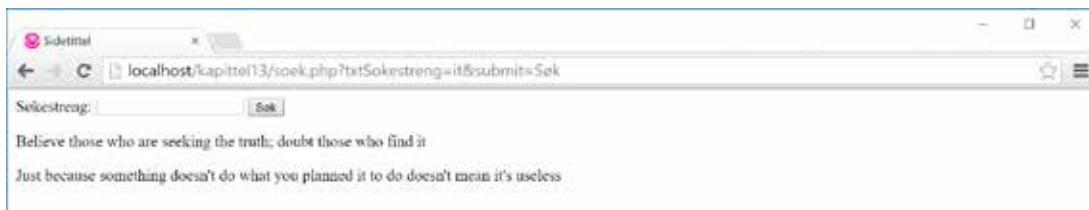


7. Lag utlistingen av sitatene som ble funnet i datasettet. Ettersom datasettet kun blir generert dersom *submit*-knappen er trykket, må vi legge til en ekstra test på at datasettet faktisk eksisterer

```
<body>
    <form action="" method="get">
        <label for="txtSokestreng">Søkestreng:</label>
        <input type="text" name="txtSokestreng" id="txtSokestreng" />
        <button type="submit" name="submit">Søk</button>
    </form>
    <?php if(isset($datasett)) while($rad = mysqli_fetch_array($datasett)) { ?>
    <p><?php echo $rad["tekst"]; ?></p>
    <?php } ?>
</body>
```

8. Test nettsiden. Kontroller at den først ikke viser noen sitater, og gjør deretter et søk etter «it» som da vil vise et begrenset utvalg sitater som inneholder denne søkestrengen

Søkestreng:



Mer om skjemaer

Svært mange dynamiske nettsider vil kreve en form for input fra brukeren. Skal vi f.eks. lage en søkerboks vil det være fornuftig at brukeren kan skrive inn tekst, eller om vi skal lage et påmeldingsskjema kan det være fornuftig å kunne velge kjønn blant to ferdigdefinerte valg.

Alt dette gjøres i HTML ved hjelp av noe som kalles for et skjema. For å lage et skjema starter vi med å skrive en `<form>`-tagg. Denne må ha en `method`-attribut der `post` og `get` er de vanligste valgene.

```
<form method="post">  
</form>
```

Metoden `post` vil gjøre at vi ikke kan sende inn skjemaet på nytt ved å oppdatere nettsiden eller dersom nettsiden blir bokmerket. Det ville f.eks. vært rart om samme ordre ble bestilt flere ganger ved å trykke oppdater på bekreftelsessiden.

Metoden `get` vil derimot legge til verdiene fra skjemaet som en del av nettadressen, og hver gang nettsiden blir besøkt eller oppdatert blir handlingen utført på nytt.

Huskeregelen blir derfor at dersom verdiene i skjemaet skal benyttes til å endre data i systemet (f.eks. legge til, oppdatere eller slette data i databasen) bør `post` benyttes. Dersom verdiene ikke gjør noen endringer (f.eks. henter ut data) bør `get` benyttes. Av den grunn benyttet vi `get` i søkerboks-eksempelet i dette kapittelet.

Skjemaet blir først nyttig når vi legger inn såkalte skjemaelementer. Vi skal i denne boka ta for oss tekstbokser, tekstfelt, ledetekster, nedtrekkslister og knapper.

Tekstbokser

Tekstbokser er kanskje et av de mest anvendelige skjemaelementene vi har. I sin enkleste form må vi sette `type` til å være `text`, og gi den et `name`.

```
Ditt navn: <input type="text" name="txtNavn" />
```

Det er navnet vi gir tekstboksen gjennom attributtet `name` som vi benytter når vi henviser til tekstboksen fra PHP-koden. Benyttes `post` som metode i skjemaet skriver vi `$_POST` i PHP, benyttes `get` som metode skriver vi `$_GET`:

```
$_GET["txtSøkestreng"]
```

Det er ikke nødvendig, men det å sette til prefikset `txt` foran navnet kan ofte lage mer lettles kode.



Pass på at navnet ikke inneholder mellomrom eller spesialtegn. Du bør også unngå de norske tegnene æ, ø og å.

PHP er case-sensitivt. Vi må derfor angi navnet på samme måte med store og små bokstaver både i definisjonen av skjemaelementet og i PHP-koden.



For å forenkle input til tekstbokser, finnes det en rekke ulike input-typer i tillegg til **text** som settes gjennom attributtet **type**. Fra PHP vil alle disse fungere likt, og gi deg en ren tekst, men utseendet og funksjonaliteten for brukerene vil være forskjellige. De mest aktuelle input-typene foreløpig vil være:

color	Velg farge: <input type="color"/>	Lar brukeren velge en farge som blir til en heksadesimal fargekode.
date	Ønsket avreisedato: <input type="date"/>	Lar brukeren velge en dato ut fra en kalender, og gjør begrensninger på hvilke verdier som er en gyldig dato.
datetime-local	Ønsket avreisedato: <input type="datetime-local"/>	Fungerer som <i>date</i> , men lar brukeren også sette et tidspunkt.
e-mail	E-post: <input type="email"/>	Lar brukeren angi en e-postadresse.
number	Alder: <input type="number"/>	Lar brukeren skrive inn tallverdier. Verdien kan også justeres ved hjelp av såkalte <i>spin-buttons</i> .
password	Passord: <input type="password"/>	Kamuflerer input fra brukeren.
range	Temperatur: <input type="range"/>	Lar brukeren velge verdi på en skala. Krever bruk av egenskapene max og min , som forklares om litt. Hva som står i hver ende av skalaen, er ren tekst før og etter skjemaelementet.
time	Velg starttid: <input type="time"/>	Lar brukeren angi et tidspunkt.
url	Velg nettadresse: <input type="url"/>	Lar brukeren angi en URL.



På mobile enheter vil tastaturet endre seg basert på hvilken input-type vi benytter.



Eksakt hvordan skjemaelementene ser ut og fungerer styres av nettleseren. Ikke alle nettlesere støtter alle input-typene, og viser da en ordinær tekstboks i stedet.



Selv om input-boksene begrenser muligheten for feil input kan de ikke forhindre bevisst manipulering. All slik feilkontroll mot hacking må derfor gjøres i PHP på serveren, og ikke i klienten.

I tillegg til at vi kan justere egenskapen **type** kan vi også endre mange andre egenskaper. De viktigste er:

maxlength

Maksimalt antall tegn som tekstboksen skal tillate, f.eks:

```
<input type="text" id="txtBrukernavn" maxlength="8" />
```

max, min og step

For typene **range** og **number** kan vi sette hva største og minste verdi skal være gjennom egenskapene **max** og **min**. Vi kan også sette hvor store steg endringen skal medføre gjennom egenskapen **step**:

```
<input type="range" name="points" min="0" max="100" step="5" />
```

placeholder

Angir en hjelpetekst som skal vises i tekstboksen inntil brukeren skriver inn sin egen tekst. Dette er et alternativ til etiketter som vi skal se på senere.

```
<input type="text" id="txtNavn" placeholder="Navn" /> <br />
<input type="text" id="txtTelefon" placeholder="Telefonnummer" />
```

Navn
Telefonnummer

value

Value refererer til innholdet i tekstboksen. Om dette settes gjennom HTML, vil det bli en slags standardverdi som vises ved lasting av nettsiden.

```
<input type="text" id="txtNavn" value="Ole Olsen" />
```

Forskjellen mellom **placeholder** og **value** er at **placeholder** viser en hjelpetekst som forsvinner når brukeren skriver inn sin egen tekst, mens **value** er en faktisk verdi i tekstboksen.



Tekstområder

En annen variant av tekstbokser er *tekstområder*. Dette er bokser der vi kan skrive flere linjer, og formatering slik som tabulatorer og ny linje kan benyttes.

Skriv melding her:

Et tekstområde lages ved hjelp av taggen **<textarea>**.

```
<textarea name="txtMelding" rows="5" cols="50">  
</textarea>
```

Ut over egenskapene **rows** for antall rader som skal vises av gangen, og **cols** for antall tegn som skal vises i hver rad kan vi også sette **placeholder** og **maxlength** på samme måte som en tekstboks. En eventuell startverdi settes mellom åpnings- og slutt-taggen.



Begrepet *tekstfelt* blir ofte benyttet både om *tekstbokser* og *tekstområder*.

Nedtrekkslister

En nedtrekksliste benyttes til å vise en liste av valg. Nedtrekkslister bygges opp av et **<select>**-element og flere **<option>**-elementer. Mye på samme måten som vi bygger opp en ordinær liste ved hjelp av **** og ****.

```
<select name="lstKunder">  
    <option value="34212">Ole Olsen</option>  
    <option value="5514">Per Persen</option>  
    <option value="21345">Nils Nilsen</option>  
</select>
```

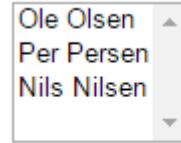
 ▾

Det er vanlig å gi navnet på nedtrekkslister prefikset *lst*.

Egenskapen **value** på hvert **<option>**-element inneholder en verdi som brukeren ikke ser, men som vi kan benytte videre i PHP-koden vår for det aktuelle valget.

Nedtrekkslister viser som standard kun ett element av gangen, unntatt under selve valgprosessen. Vi kan påvirke dette ved å endre egenskapen **size**. Endres denne til noe annet enn 1, vises det i stedet en såkalt listeboks, der flere elementer er synlige av gangen.

```
<select id="lstKunder" size="4">
    <option value="34212">Ole Olsen</option>
    <option value="5514">Per Persen</option>
    <option value="21345">Nils Nilsen</option>
</select>
```



Ledetekster

Foran skjemaelementer kan vi som vi har sett skrive helt ordinær tekst og HTML-kode:

```
Ditt navn: <input type="text" name="txtNavn" />
```

Det er imidlertid vanlig å lage slike ledetekster ved hjelp av **<label>**-taggen. Ettersom **<label>**-taggen har et **for**-attributt som skal peke til **id** på det elementet det er ledetekst for, må vi også legge til en **id** i **<input>**-taggen. Denne kan være lik **name**.

```
<label for="txtNavn">Ditt navn:</label>
<input type="text" name="txtNavn" id="txtNavn" />
```

Det kan i utgangspunktet virke tungvint å benytte **<label>** for å oppnå eksakt det samme som en ren tekst. Det er allikevel viktig, spesielt for tilgjengelighet. En skjermleser vil f.eks. enklere kunne lese opp og la brukeren navigere i et skjema dersom vi benytter **<label>**.

Knapper

For å kunne utføre en handling når brukeren er ferdig med å fylle ut skjemaet må vi ha en *submit*-knapp inne i **<form>**-taggen.

Denne knappen settes inn med følgende kode:

```
<button type="submit" name="submit">Søk</button>
```

Det vi angir som **value** er det som vil vises som tekst på knappen.



Det er også mulig å lage en *reset-knapp* som tømmer skjemaet for data. Denne settes inn slik:

```
<button type="reset">Tøm skjema</button>
```

Siden en reset-knapp ikke skal refereres til fra PHP, trenger ikke denne noe **name**-attributt.

TIPS