



Avdeling for informatikk og e-læring, Høgskolen i Sør-Trøndelag

## JavaScript

Svend Andreas Horgen

11.09.2007

Lærestoffet er utviklet for faget LV357D Webteknikker

## 5. JavaScript (fra leksjon 5 i faget Webteknikker)

*Resymé: JavaScript foregår på klienten, men det er likevel ofte bruk for JavaScript i webprogrammering (som PHP eller ASP.NET). JavaScript utfyller som regel webbaserte teknologier, og er også primærteknologien i Ajax-drevne applikasjoner. Målet med denne leksjonen er å gi innblikk i mulighetene med JavaScript og en kort introduksjon til språket. Vi holder oss på et overordnet nivå.*

5.1.	HVA ER JAVASCRIPT?.....	2
5.2.	SYNTAKS OG EKSEMPLER PÅ KODE.....	3
5.2.1.	Eksempel på funksjoner i JavaScript.....	3
5.2.2.	Hendelser.....	4
5.2.3.	Grunnleggende syntaks .....	5
5.2.4.	Validere et skjema med JavaScript.....	7
5.3.	LITT MER AVANSERT BRUK AV JAVASCRIPT .....	10
5.3.1.	DOM – Document Object Model .....	11
5.3.2.	Mer DOM ved hjelp av metoden getElementById() .....	11
5.3.3.	Et genialt verktøy – FireBug .....	13
5.3.4.	Bruk av ferdige script .....	14
5.3.5.	Samspill mellom JavaScript og andre teknologier .....	18

## 5.1. Hva er JavaScript?

JavaScript ble utviklet av Netscape i 1995, og har fått stor utbredelse på web. De fleste nettlesere støtter JavaScript, men det er som regel mulig for brukeren å slå av JavaScript. Dette er viktig å ha i bakhodet når en utvikler websider med JavaScript, siden brukere som har nettlesere hvor JavaScript er avslått eller ikke støttes, ikke vil se noe resultat av JavaScript.

Mange tror at JavaScript er det samme som Java, eller i det minste en nær slektning. Det stemmer ikke. Språkene har nokså lik syntaks, men ellers store forskjeller. En trenger ikke deklare datatyper i JavaScript, og JavaScript-kode tolkes (mens Java-kode kompiles til en kjørbart fil). Det betyr at når et script har en feil i linje 13, kan meget mulig linje 1-12 kjøres før scriptet stopper. JavaScript er også enkelt å integrere i HTML-kode. Det er rett og slett bare å plassere en script-tag et sted i HTML-koden, og legge JavaScript-kode inne i denne script-taggen. Alt dette gjør terskelen for å komme i gang lav, og dette er nok en viktig årsak til at JavaScript er så utbredt. JavaScript mangler for øvrig arv og klasser, men er likevel objektbasert til en viss grad.

Mange utviklere anser JavaScript som et for enkelt scriptspråk til å kunne utføre profesjonell utvikling. Google og andre store aktører har med sin satsning på Ajax generelt og JavaScript spesielt, fått utviklere verden over til å revurdere gamle fordommer mot språket.

JavaScript anses ofte for å være et onde på web, siden mange websider bruker JavaScript for å åpne vinduer (popup) med reklame og liknende. Resultatet er at nyere nettlesere tilbyr å blokkere kjøring av script eller popup-vinduer, og mange assosierer JavaScript med virus – feilaktig. Det er riktignok en viss fare for at de som tillater JavaScript i nettleseren kan kjøre kode som stjeler cookies og annen informasjon, eller i verste fall aktiverer skummel programkode, men i denne leksjonen skal vi fokusere på hva JavaScript (kan/bør) brukes til.

Det er liten tvil om at JavaScript spiller en viktig rolle for mange nettsted i dag, og vi skal se hva JavaScript er, oppsummere syntaksen kort (slik at du skal kunne lese kode og forstå sann ca hva som skjer), og vise hvordan ferdige script som andre har laget, kan integreres på ditt nettsted.

Noe av det fine med JavaScript, er at det gradvis kan bygges inn i et nettsted. Her er noen av de tingene du kan bruke JavaScript til:

- Validere om et skjema er riktig utfylt før det sendes til brukeren.
- Navigere i historikken, for eksempel gå tilbake til forrige side som brukeren var på, ved å trykke på en knapp eller en lenke.
- Finne ut mer informasjon om hvilken nettleser brukeren har.
- Vise og skjule informasjon, noe som muliggjør for eksempel å lage menyer som åpner/lukker seg.
- Utføre beregninger på klientsiden, uten at tjeneren må belastes (små kalkulatorer).
- Lage mer brukervennlige websider, med for eksempel klikk-og-dra-løsninger og andre gui-elementer som er kjent fra vanlige applikasjoner.
- Som bruker benytte et webbasert skrive/regnearkprogram, for eksempel med "Google Docs & Spreadsheets" ([www.google.com](http://www.google.com)).

## 5.2. Syntaks og eksempler på kode

Når du skal bruke JavaScript i websiden din, er det bare å skrive kode i script-taggen:

```
<html>
<head><title>Første JavaScript</title></head>
<body>
<h1>En liten JavaScript-test</h1>
<script language="javascript">
    alert("Tjänare, grabben, Hej Tellus");
</script>
</body>
</html>
```

Koden som ligger i script-taggen kjøres når script-taggen tolkes, såfremt JavaScript er slått på i nettleseren. Du kan ha så mange script-tagger du vil i dokumentet. Koden kjøres i det øyeblikk script-taggen møtes på. En `alert()` viser en meldingsboks. Vi skal nå se på JavaScript-syntaks gjennom noen eksempler.

### 5.2.1. Eksempel på funksjoner i JavaScript

I JavaScript er det ikke nødvendig å avslutte setninger med semikolon, men det er en god vane siden en i nesten alle andre språk bruker semikolon for å avslutte setninger (gammel vane er vond å vende etc).

Dersom du skal lage dine egne funksjoner, bør disse defineres i head-seksjonen, og så kalles opp, slik:

```
<html>
<head><title>Eksempel på funksjoner</title>
    <script language="javascript">
        function visMelding() {
            alert("Tjänare, grabben, Hej Tellus");
        }
    </script>
</head>
<body>
<h1>En liten JavaScript-test</h1>
<p>Her er en side med JavaScript<br />
<a href="javascript:visMelding();">Klikk denne lenken</a> for å vise en
melding. <br />
Du kan også klikke på denne knappen <input type="button"
onclick="visMelding();" value="Vis en melding" /> for å vise en melding
</p>
</body>
</html>
```

Dette eksempelet er illustrert i Figur 1. Legg merke til hvordan samme JavaScript-funksjon kan kalles fra flere steder i koden, både fra en lenke (ved å angi javascript: som prefiks i href-verdien), og ved klikk på en knapp.



Figur 1: Når lenken eller knappen klikkes, kalles en JavaScript-funksjon opp som viser en melding.

Det er selvsagt også mulig å sende argumenter til funksjoner som i andre språk.

### 5.2.2. Hendelser

I koden over så du hvordan en JavaScript-funksjon var knyttet opp mot onclick-hendelsen til en knapp. Det er også mulig å knytte andre hendelser opp mot knapper (og andre elementer), og de fleste er like selvforklarende som onclick (klikk med museknapp).

Musehendelser:

`onclick/onclick`: enkelt- eller dobbeltklikk

`onmousedown/onmouseup`: museknapp klikkes ned/opp, atomiske operasjoner

`onmousemove`: mus beveger seg

`onmouseover/onmouseout`: musepeker kommer over et element/bort igjen

Tastaturhendelser:

`onkeydown`: trykk på tastaturknapp

`onkeypress`: så lenge en tastaturknapp holdes nede

`onkeyup`: når en nedtrykt knapp slippes

Diverse hendelser:

`onabort`: noe avbrytes

`onfocus/onblur`: fokus til eller bort fra et element, for eksempel tekstfelt

`onchange`: noe endres, for eksempel et tegn skrives inn i et tekstfelt

`onerror`: når noe har gått galt

`ondragdrop`: et element dras og slippes

`onload`: siden lastes  
`onunload`: nettleseren går til en ny side  
`onreset`: Reset-knapp klikkes  
`onresize`: brukeren endrer størrelse på et element eller en side  
`onsubmit`: skjemainformasjon sendes, submit-knappen trykkes

Det er ikke bare knapper i et skjema som kan gjøre nytte av slike hendelser. Alle HTML-elementer kan benytte hendelsene.

### 5.2.3. Grunnleggende syntaks

Det er ikke nødvendig å deklare variabler i JavaScript, men det er lov å bruke ordet `var` foran variabler ved opprettelse. Dette er lurt å benytte seg av, fordi koden da blir lettere å lese. Du trenger heller ikke tenke på datatyper når du koder i JavaScript.

Neste eksempel illustrerer syntaksen, og trenger neppe nærmere forklaring utover kommentarene.

```
alder = 28; //lov, men best å ha "var" foran nye variabler
var enAnnenAlder = 45;
var navn = "Ola Nordmann"; //både anførselstegn "...
var etAnnetNavn = 'Kari Nordmann'; //... og fnutter ' kan brukes i strenger
var sum = alder + enAnnenAlder; //lagrer 73 i variabelen sum
navn = "Petter Nordmann"; //ikke var her, variabel er deklartert fra før
var rik = false; //boolsk variabel
alder++; //alder er nå 29
enAnnenAlder -= 10; //nå lik 35
navn = 23; //variabelen navn skifter nå datatype fra string til integer
```

Med `document.writeln()` kan en vanlig linje skrives ut i nettleseren (ikke i en alert-boks, men som del av den teksten som brukeren ser). Dette kan brukes for å vise en spesiell farge på søndager:

```
<html>
<head><title>Søndager vises med annen farge</title></head>
<body>
<script language="javascript">
    var d = new Date(); //finner informasjon om nåværende dato og tidspunkt
    var dag = d.getDay();
    //alert(dag); //fjern kommentaren for å teste hvilken dag det er
    //dag = 0; //fjern kommentarene for å teste at farge vises riktig
    if (dag == 0) {
        document.writeln("<h1 style='color:green'>I dag er det søndag</h1>");
    }
    else {
        document.writeln("<h1 style='color:black'>I dag er det
arbeidsdag</h1>");
    }
</script>
</body>
</html>
```

Her ser du også eksempel på en if-else-struktur med JavaScript. Du ser også hvordan stilsett (CSS) brukes for å endre fargen til grønn, men her kunne vi også brukt en font-tag med color-

attributt satt, eller andre ting. Legg merke til hvordan vi med `document.writeln()` kan skrive ut vanlig tekst og HTML-kode. Nettleseren tolker HTML-koden, og dermed vil ”I dag er det søndag” vises som en h1-overskrift dersom det er søndag i dag.

I JavaScript er det slik at et Date-objekt har mye informasjon om nåværende dato og tidspunkt. Med metoder og egenskaper, kan vi plukke ut informasjon fra dette dato-objektet. Metoden `getDay()` henter ut et tall som representerer dagen i dag. Tallet 0 betyr søndag, mens 6 betyr lørdag. Dermed kan vi i betingelsen i if-testen sjekke om brukeren besøker websiden på en søndag eller en vanlig ukedag. Legg merke til at vi bruker `==` for å teste på likhet, mens `=` for å tilordne verdier til en variabel.

Neste eksempel viser hvordan en løkke kan brukes til å skrive ut alle ukedager. Først lages en matrise (også kalt tabell, og på engelsk array). Innholdet er tekststrenger som utgjør de norske navnene på ukedagene. Deretter finner vi dagens datoinformasjon, og plukker ut tallet som representerer denne dagen ved hjelp av metoden `getDay()`. Deretter skriver vi ut ”I dag er det tirsdag” dersom det er tallet 2 som ligger i variabelen som heter `idag`. Legg merke til hvordan vi benytter oss av at matrisen har samme indeks som første mulige dag. Metoden `getDay()` returnerer et tall som er mellom 0 og 6, og matrisen har 7 elementer, med indekser (nøkler) fra 0 til 6. Det er samsvar, og vi kan derfor bruke returen fra `getDay()` til å hente fram riktig dagsnavn på norsk!

```
<html>
<head><title>Hvilken dag er det i dag, hvilken ...</title></head>
<body>
<script language="javascript">
    var ukedag=new Array("Søndag",
                        "Mandag",
                        "Tirsdag",
                        "Onsdag",
                        "Torsdag",
                        "Fredag",
                        "Lørdag"
                        );

    var d = new Date();
    var idag = d.getDay();
    document.write("<p>I dag er det " + ukedag[idag] + "</p>");

    document.write("<p>Alle ukens dager, er: </p>");
    document.write("<ul>");
    for (teller=0; teller < ukedag.length; teller++){
        document.write("<li>" + ukedag[teller] + "</li>");
    }
    document.write("</ul>");
</script>
</body>
</html>
```

I for-løkken skrives alle dagene ut. Vi lar for-løkken gå fra 0 til 6, men i stedet for å hardkode tallet 6, henter vi antall elementer i matrisen (som er 7) ved hjelp av egenskapen `length` anvendt på matrisen. Figur 2 viser at dagene skrives ut som en punktmerket liste. Overbevis deg selv om at dette er tilfelle (forstå koden).



Figur 2: Med en løkke kan elementer fra en matrise (array) enkelt brukes og skrives ut.

Det fins også en rekke nyttige metoder for å behandle strenger, for eksempel `toLowerCase()`, `indexOf()` og `substring()`. Vi kommer tilbake til noen av disse i neste eksempel.

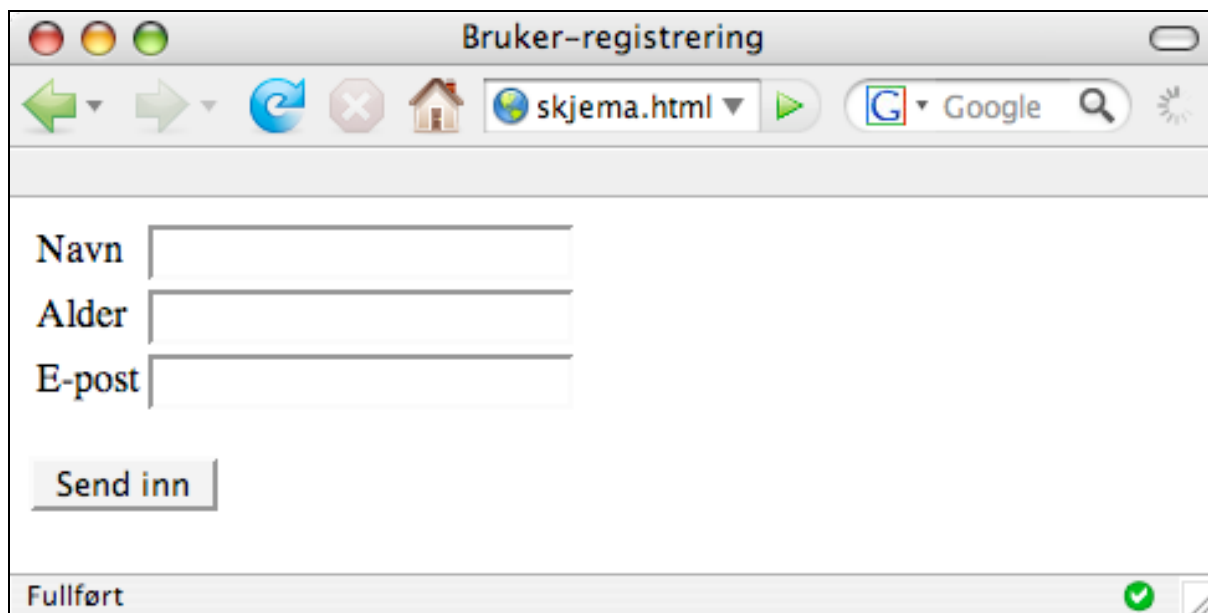
#### 5.2.4. Validere et skjema med JavaScript

Er skjemaet som brukeren har lagt tid og flid i å fylle ut, riktig utfylt? Kan det være at det mangler en alfakrøll i e-postadressen? I så fall er det uheldig om skjemaet blir sendt til tjeneren uten at brukeren får anledning å rette opp i feilen først.

! All input fra brukeren skal (hvis en har fokus på sikkerhet) valideres på tjeneren. Det er derimot ofte lurt å også validere på klienten. Et JavaScript på klienten utføres

- umiddelbart, og dermed blir gjerne brukeropplevelsen bedre. Med klientside-validering trenger ikke brukeren å vente på at tjeneren oppdager feilen og at en ny side skal lastes på nytt. I tillegg sparer man tjeneren for unødvendig belastning, og det blir ikke unødvendig trafikk over nettverket.

Gitt følgende skjema:



Figur 3: Et fint formattert skjema. Dette bør valideres når knappen trykkes.

HTML-koden for å lage skjemaet er rett fram. Vi har her plassert ledetekstene og input-feltene i to ulike kolonner i en tabell, for å få et fint skjema. Legg merke til at skjemaet heter *mittSkjema*, og at vi har et felt som heter *alder* og et som heter *epost*.

```
<html>
<head><title>Bruker-registrering</title>
<script type="text/javascript" language="javascript">
    //JAVASCRIPT-KODE FOR Å VALIDERE KOMMER HER
</script>
</head>
<body>
<form name="mittSkjema"
    action="behandle.php"
    method="post"
    onSubmit="return validerSkjema();"
>
<table>
    <tr>
        <td>Navn</td>
        <td><input type="text" name="navn" /></td>
    </tr>
    <tr>
        <td>Alder</td>
        <td><input type="text" name="alder" /></td>
    </tr>
    <tr>
        <td>E-post</td>
        <td><input type="text" name="epost" /> </td>
    </tr>
</table>
<p><input type="submit" name="knapp" /></p>
</form>
</body>
</html>
```

Når knappen trykkes, skal skjemaets data normalt sendes til scriptet `behandle.php`. Vi ønsker nå å forhindre dette, eller kanskje heller – utsette dette litt. Det gjøres ved å sette opp egenskapshåndtereren `onsubmit`. JavaScript-funksjonen `validerSkjema()` kalles opp, og dersom denne funksjonen returnerer verdien `true`, vil det i `onsubmit` stå `"return true"`, og da sendes skjemaet til `behandle.php`. Hvis funksjonen `validerSkjema()` returnerer verdien `false`, vil det i `onsubmit` stå `"return false"`, og da *avbrytes* sendingen av skjemaet.

Det vi da trenger å gjøre, er å lage en funksjon som returnerer verdien `true` dersom skjemaet er korrekt utfylt, og `false` hvis skjemaet er feil utfylt. Funksjonen ser altså slik ut i grove trekk:

```
<script type="text/javascript" language="javascript">
    function validerSkjema() {
        //if noe går galt
        return false;
        //elseif alt går bra (alt er riktig utfylt)
        return true;
    }
} //slutt på funksjonen her
</script>
```

Vi kan plassere de feilsjekker vi ønsker inne i funksjonen vår. Det er naturlig å sjekke om e-postadressen har en alfakrøll og et punktum i seg. Det er også naturlig å sjekke om alderen er mellom 0 og 100 år. Navnet foretar vi ikke noen feilsjekk på i denne omgang.

```
<script type="text/javascript" language="javascript">
    function validerSkjema() {
        var alder = window.document.mittSkjema.alder.value;
        var epost = window.document.mittSkjema.epost.value;
        if (alder < 0 ) {
            alert("alder må være større enn 0 år");
            window.document.mittSkjema.alder.focus();
            return false;
        }
        if (alder > 100 ) {
            alert("alder må være mindre enn 100 år");
            window.document.mittSkjema.alder.focus();
            return false;
        }
        if (epost.indexOf("@") < 0 ) {
            alert("e-postadressen din mangler snabel-a");
            window.document.mittSkjema.epost.focus();
            return false;
        }
        if (epost.indexOf(".") < 0 ) {
            alert("e-postadressen din må ha minst ett punktum i seg");
            window.document.mittSkjema.epost.focus();
            return false;
        }

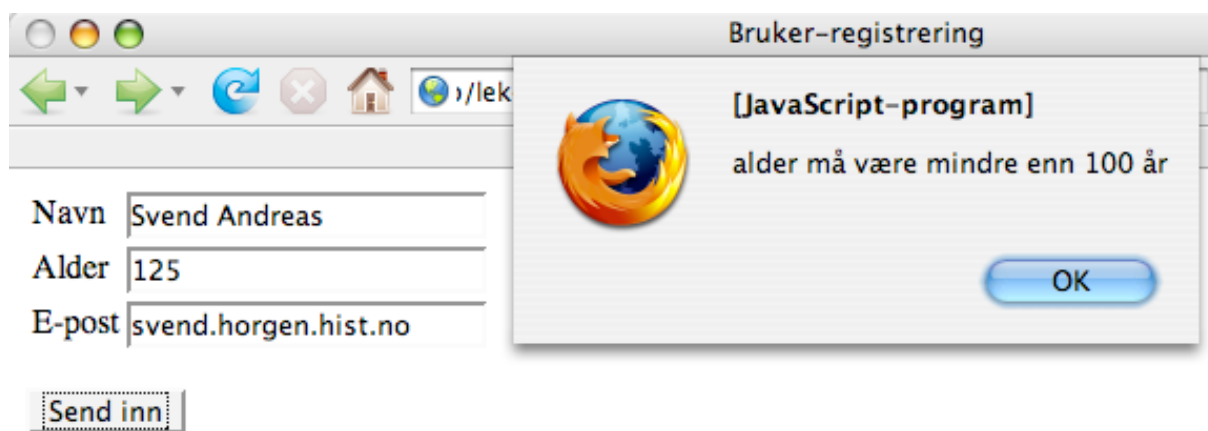
        return true; //skjer bare hvis if ikke kjøres
    } //slutt på funksjonen her
</script>
```

Med en gang JavaScript støter på kodeordet `return`, vil funksjonen avsluttes med angitt returverdi. Derfor er det kun én mulighet for å lykkes (med å sende skjemaet til `behandle.php`), nemlig hvis ingen av if-testene slår til.

Legg merke til at vi først i funksjonen henter ut den verdien som brukeren har skrevet inn i skjemaets felter for e-post og alder. Vi navigerer i DOM (document object model) og plukker ut teksten (verdien) som ligger i vinduet på websiden i skjemaet i det aktuelle tekstfeltet.

! Dersom noe går galt, så gir vi fokus til riktig tekstfelt, slik at markøren står og blinker i det feltet som har feil informasjon skrevet inn. Dermed slipper brukeren å lete etter

- hvilket tekstfelt som må fylles ut på nytt. Dette er en fin detalj. Tenk over hvorfor det ikke er så lurt å samtidig nullstille feltet som var feil utfylt. Det er mulig, men mange vil trolig irritere seg over å miste informasjonen de har brukt tid på å fylle ut.



Figur 4: JavaScript sjekker om skjemaet er feil utfylt, og viser i så fall feilmelding.

I koden for valideringen av skjemaet, ser du også bruk av streng-metoden `indexOf()`. Denne sjekker rett og slett om angitt tegn eksisterer i tekststrengen (som kommer før punktumet), og dersom den gjør det, returneres posisjonen til første forekomst av tegnet. Hvis den ikke fins, returneres `-1`. Dermed er det lett å sjekke om e-postadressen har et nogenlunde ordentlig format. Vi sjekker på ingen måte om e-posten er helt korrekt utfylt, men dersom brukeren glemmer krøllalfa eller punktum, så er i hvertfall noe galt. Det kan være greit å overlate mer grundig feilsjekking til tjeneren. Det er for øvrig umulig å validere at e-postadressen faktisk eksisterer og ikke bare er på riktig form, men det er en annen diskusjon.

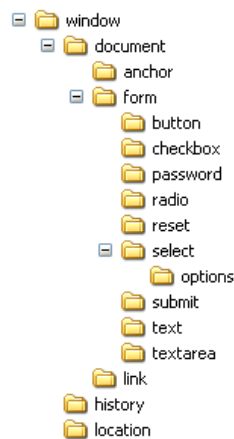
### 5.3. Litt mer avansert bruk av JavaScript

Det er nå på tide å gå litt mer i dybden på virkemåten til JavaScript. Du kan selvsagt ikke lære noe særlig grundig å programmere JavaScript på bare en leksjon, og i stedet for å gå i dybden på programmering, viser vi enda mer av det totale oversiktsbildet. Det er nyttig å forstå hva DOM er, så vi starter med det. Videre er det kjekt å vite noe om hvordan du kan integrere andres JavaScript-kode på dine egne websider.

### 5.3.1. DOM – Document Object Model

Et HTML-dokument har en viss struktur, med overskrifter, avsnitt, tekstlig innhold og så videre. DOM står for Document Object Model og er en hierarkisk representasjon av alle strukturelle elementer på en HTML-side. JavaScript kan operere mot nodene i DOM-treet, for eksempel legge til, endre, slette, vise/skjule og liknende. Dermed er det mulig for klienten å endre innholdet på en webside etter at den er mottatt fra tjeneren.

Alle HTML-elementer blir en del av sidens DOM når siden vises i nettleseren. Sagt med andre ord: Nettleseren har en DOM-beskrivelse av en nettside. Dersom nettleseren har flere nettsider åpne (ulike arkfaner eller vinduer), har hver side en egen DOM med ulikt innhold.



Figur 5: Et utdrag av DOM-hierarkiet.

Som vi så i eksempelet med validering av skjemaet, var det mulig å navigere seg fram til verdien i et tekstfelt. Mer nøyaktig skrev vi setningene (på ulikt sted i koden)

```
<form name="mittSkjema">
<input type="text" name="epost" />
var epost = window.document.mittSkjema.epost.value;
```

Sammenlikn disse tre kodelinjene med den hierarkiske representasjonen av en generell DOM fra Figur 5, og du vil se at vi kan navigere ved å navngi skjemaet `mittSkjema` og tekstfeltet `epost`, og deretter referere til egenskapen `value`.

### 5.3.2. Mer DOM ved hjelp av metoden `getElementById()`

Det er også mulig å unngå navigasjon på denne måten i DOM-treet ved å bruke id-attributtet. Som vist i neste eksempel, har vi laget et div-element med `id="boksenMin"`. Denne boksen kan vi henviser til med JavaScript-kode ved å bruke `getElementById()` metoden til document-objektet. I det vi åpner websiden, ser vi en ramme rundt teksten.



Figur 6: I utgangspunktet vises en ramme rundt teksten...

Når vi beveger musa over teksten, vises derimot en tykk, prikket linje som ramme.



Figur 7: ... Når musepekeren kommer over "boksen" vil rammens fasong endre seg.

Som vist i koden som realiserer denne dynamiske løsningen, plukker vi ut et bestemt element i DOM-treet til nettleseren, og manipulerer dette på direkten.

```
<html>
<head><title>Eksempel på funksjoner</title>
<script language="javascript">
  function endreRamme(){
    var boks = document.getElementById("boksenMin");
```

```

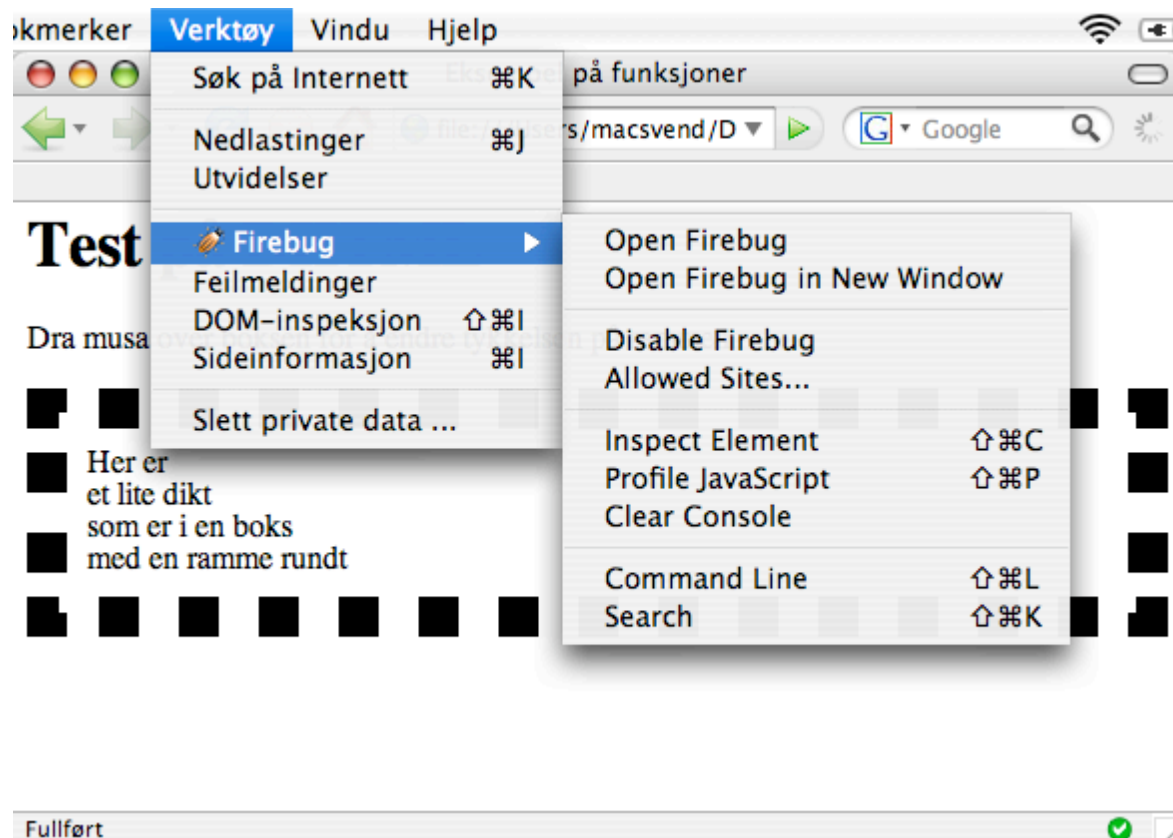
        boks.style.border = "20px dotted";
    }
</script>
</head>
<body>
<h1>Test på DHTML</h1>
<p>Dra musa over boksen for å endre tykkelsen på rammen</p>
<div id="boksenMin"
    style="border-width:1px; border-style:solid; padding:10px"
    onmouseover="endreRamme ();">
    Her er <br />
    et lite dikt <br />
    som er i en boks <br />
    med en ramme rundt
</div>

</body>
</html>

```

### 5.3.3. Et genialt verktøy – FireBug

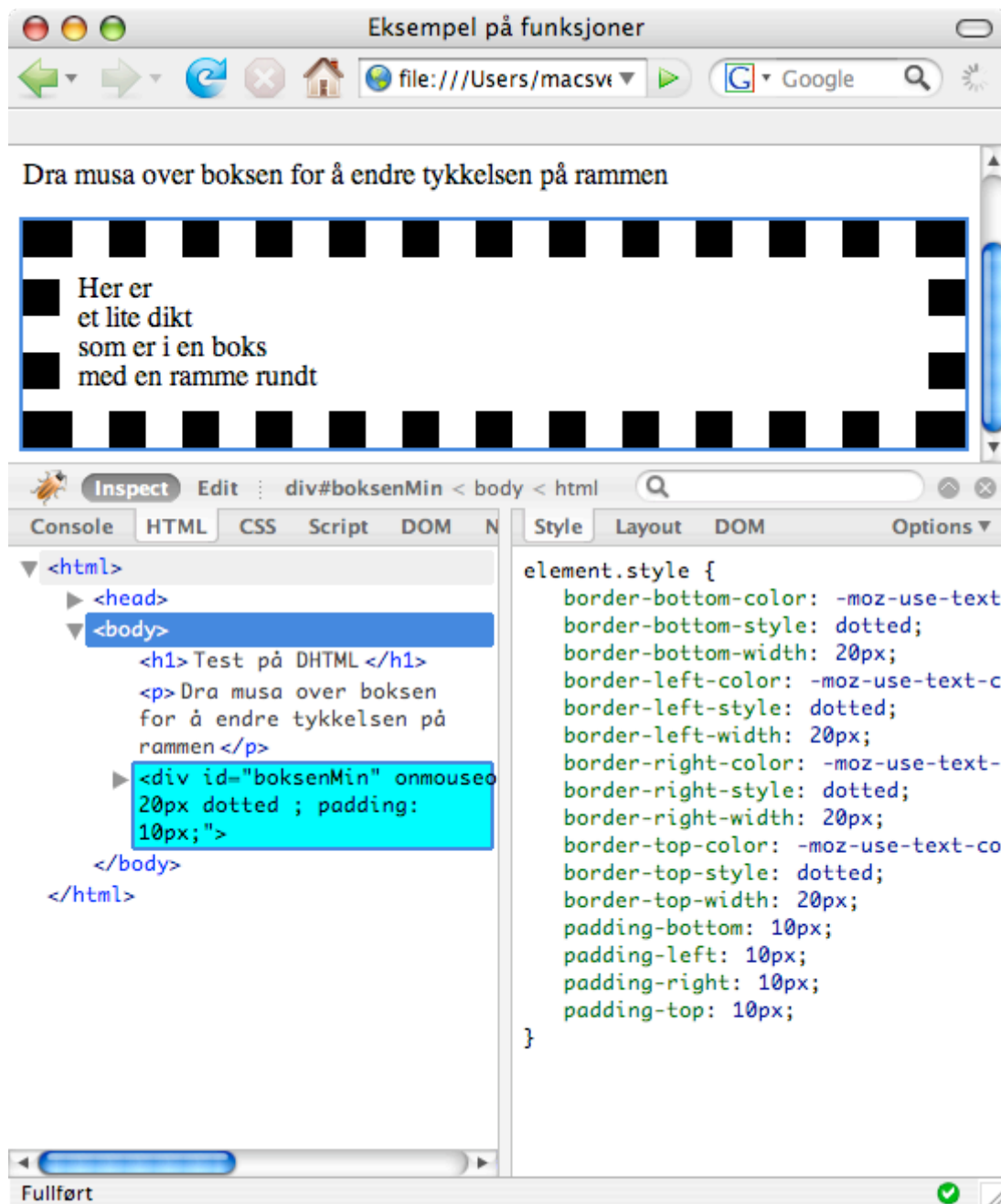
Mozilla FireFox har en flott utvidelse som heter FireBug som lar deg som bruker se nærmere på DOM-treet i nettleseren.



Figur 8: Utvidelsen FireBug (for Mozilla FireFox) er enhver webutviklers venn.

Med FireBug kan du ikke bare se på DOM-treet på en oversiktlig måte, men du kan også endre på DOM-treet real-time! Det vil si at du kan besøke en hvilken som helst webside og endre i DOM-treet, og se hvordan siden vil se ut dersom selve kildekoden blir endret.

Hensikten er selvsagt å gjøre det enklere og mindre tidkrevende å finne feil, siden du ikke trenger å oppdatere HTML-koden hvis du bare skal teste ala ”hva skjer hvis jeg endrer til 20px i stedet for 1px”.



Figur 9: Ved å bevege musepekeren over div-elementet, vil FireBug markere DOM-elementet med blått og stil-informasjon til høyre. Legg merke til at det er en rekke ”knapper” som kan trykkes på for å se ulik type informasjon, og også en ”Edit”-knapp for å endre på både tekstlig innhold, kode og DOM-treet (real time). Rett og slett en fantastisk plugin til FireFox!

#### 5.3.4. Bruk av ferdige script

Noe av det fine med JavaScript, er at det (i likhet med HTML) er lett å lære av andre, og lett å integrere andres løsninger i sin egen. Dersom du finner en webside som har en fin

menyløsning hvor menyen lukker og åpner seg, så er det bare å se på kildekoden og enten bli inspirert eller klippe og lime. JavaScript må sendes til nettleseren for å kunne kjøres, og derfor er all kode tilgjengelig til inspeksjon.

Det samme er *ikke* tilfelle med PHP/ASP.NET. Dersom du for eksempel går til et PHP-basert nettsted, så ser du *resultatet* av PHP-scriptet, altså ikke selve PHP-scriptet selv. Det betyr ikke at PHP er dårligere eller vanskeligere å bruke enn JavaScript, men rett og slett at den ene (primært) er en klientteknologi mens den andre er en tjenerteknologi.

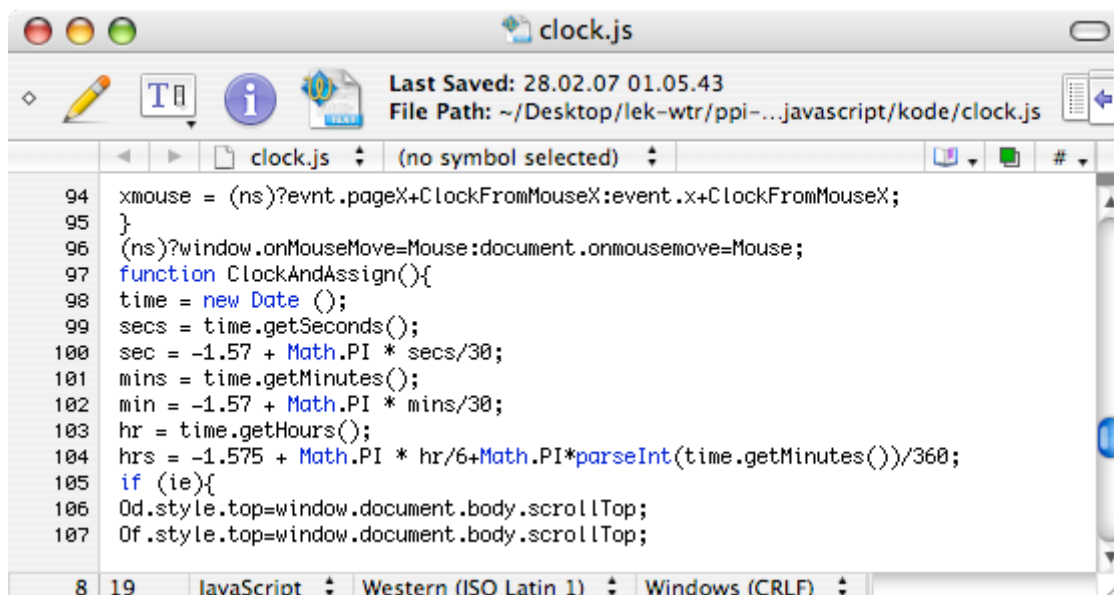
Ofte vil en JavaScript-løsning bestå av hundrevis av kodelinjer. I stedet for å få en lang HTML-fil, er det vanlig å plassere koden i funksjoner i en egen fil, og lagre filen med navn på formen *filnavn.js*. En kan inkludere slike eksterne filer (på samme måte som en kan inkludere eksterne stilsett) ved å plassere følgende setning i head-seksjonen til en HTML-side:

```
<script src="filnavn.js" type="text/javascript"></script>
```

Fordelene ved å bruke eksterne JavaScript-filer, er mange:

- All kode som naturlig hører sammen ligger lagret på ett sted.
- Koden blir mer ryddig siden logikk og design er adskilt (JavaScript og HTML er ikke mikset sammen).
- Mange sider kan bruke samme eksterne JavaScript-fil.
- Ved å endre kode på ett sted vil endringene gjelde for alle websider som lenker inn den eksterne filen.

Vedlagt denne leksjonen ligger en morsom klokke som viser tiden på en original måte. Koden ligger i en egen JavaScript-fil som heter clock.js. Denne er hentet et sted på Internett (for mange år siden, så kilde er ukjent).



```
94  xmouse = (ns)?evnt.pageX+ClockFromMouseX:event.x+ClockFromMouseX;
95  }
96  (ns)?window.onMouseMove=Mouse:document.onmousemove=Mouse;
97  function ClockAndAssign(){
98  time = new Date ();
99  secs = time.getSeconds();
100 sec = -1.57 + Math.PI * secs/30;
101 mins = time.getMinutes();
102 min = -1.57 + Math.PI * mins/30;
103 hr = time.getHours();
104 hrs = -1.575 + Math.PI * hr/6+Math.PI*parseInt(time.getMinutes())/360;
105 if (ie){
106 Od.style.top=window.document.body.scrollTop;
107 Of.style.top=window.document.body.scrollTop;
```

Figur 10: Utdrag fra koden til clock.js – det er en meget dyktig utvikler som har laget koden.

Selv om koden til clock.js har totalt 156 linjer med kode, vises ikke dette i HTML-filen. Vi lenker ganske enkelt inn med en eneste setning (her er riktignok setningen brutt over flere linjer for å øke lesbarheten) :

```
<html>
<head>
  <script type="text/javascript"
    language="javascript"
    src="clock.js"
  ></script>
</head>

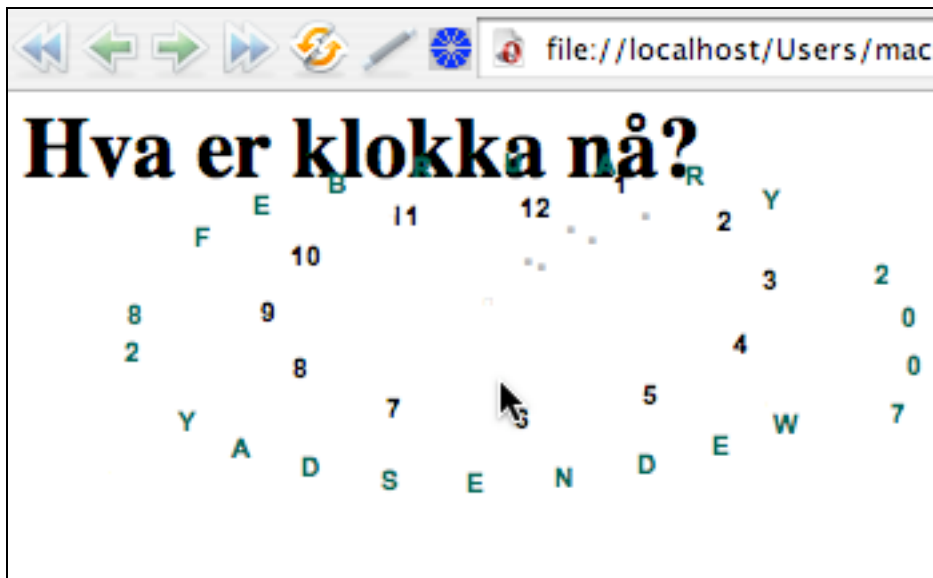
<body bgcolor="white">
<h1>Hva er klokka nå?</h1>
</body>
</html>
```

Resultatet blir at klokka vises på websiden for de som måtte komme på besøk:



Figur 11: Klokka som resultat av clock.js har en dato som roterer rundt, og visere som viser nøyaktig tidspunkt. Sekundviseren tikker og går den også.

Det ekstra morsomme med denne klokka er ikke bare at dens dato roterer rundt selve klokka, men den er også sentrert rundt musepekeren ”med forsinkelse”, slik at hvis brukeren beveger musa vil klokken sprette opp og ned som en jojo (eller gele om man vil). Dette er forsøkt illustrert i Figur 12.



Figur 12: Hvis musepekeren beveges, danser klokka opp og ned med gele-effekt, inntil den slår seg til ro et par sekunder senere.

Det er mulig å konfigurere klokka til en viss grad. De første linjene av koden i clock.js ser slik ut:

```
dCol='005544';//date colour.
fCol='000000';//face colour.
sCol='FFFFFF';//seconds colour.
mCol='AAAAAA';//minutes colour.
hCol='AAAAAA';//hours colour.
ClockHeight=50;
ClockWidth=100;
ClockFromMouseY=0;
ClockFromMouseX=0;
```

Den observante leser ser at det kun er to visere i Figur 11 og Figur 12, nemlig for timer og minutter. Den observante webutvikler, ser også nå hvorfor: Fargen på sekundviseren er satt til hvit, og bakgrunnsfargen på websiden er satt til hvit. Ved å endre `sCol='FFFFFF'` til for eksempel `sCol='000000'`, vil sekundviseren bli svart, og vises. Det er også mulig å endre dimensjonen på klokka, og å posisjonere musepekeren utenfor klokkas sentrum.

Som du ser er det enkelt å finne ut hvordan du skal tilpasse andres løsninger. Du trenger ikke mye mer kunnskap i JavaScript for å kunne ta tak i eksisterende script fra andre, tilpasse/endre og integrere med ditt eget nettsted. Muligheten for å lenke inn eksterne filer åpner for at dette er lett å få til i praksis.

Klokke-eksempelet er morsomt å se på i praksis, men det bør ikke brukes på seriøse nettsteder. Det fins enorme mengder av gratis nedlastbare JavaScript på web som du kan bruke for å piffe opp sidene dine, men en vanlig (nybegynner)feil er å overdrive. Mulighetene er der, og for mange blir den største utfordringen å beholde gangsynet. Godt design og mye JavaScript hører sjelden sammen. Her er det ingen absolutte regler, det er selvsagt mulig å lage gode løsninger som har mye JavaScript, men en bør tenke aktivt gjennom omfanget, og også ta hensyn til de brukere som eventuelt har JavaScript avslått, blokkert eller mangler støtte for JavaScript i sin nettleser.

### 5.3.5. Samspill mellom JavaScript og andre teknologier

Vi har tidligere sett på hvordan CSS kan brukes sammen med JavaScript/DOM for å endre på tykkelsen på en kantlinje rundt en boks.

Du vet nå at du kan lenke inn eksterne JavaScript, men hva om du vil velge et av flere JavaScript basert på for eksempel hvem brukeren er, eller hvilken dag det er? Kanskje har du en løsning som tilsier at brukeren skal få se en klokke bare på søndager?

PHP og JavaScript passer som fot i hanske (eller var det som hånd i hose? ;-). Teknologiene utfyller hverandre. Sjekk følgende PHP-script:

```
<html>
<head>

<?php
    if (date("D") == "Sun"){
        echo "<script type='text/javascript'
            language='javascript'
            src='clock.js'
            ></script>";
    }
    else {
        echo "<script type='text/javascript'
            language='javascript'
            src='kjedelig-klokke.js'
            ></script>";
    }
?>

</head>

<body bgcolor="white">
<h1>Hva er klokka nå?</h1>
</body>
</html>
```

Her sjekker vi først med PHP hvilken dag det er. Hvis det er søndag, kjøres if-setningen, hvis ikke kjøres else-delen. Vi velger altså hvilket JavaScript brukeren skal få tilsendt allerede på tjeneren!

Tenk videre over de mulighetene dette bringer. Du kan for eksempel ha et JavaScript for å lage en meny med en rekke menyvalg, undermenyer som kan åpnes og lukkes, og så videre. Hva skal være teksten i menyvalgene? Jo, det ligger antakeligvis i JavaScript-filen som tekst. I stedet for å hardkode dette, kan du generere innholdet med PHP. Det betyr for eksempel at du kan lage et menysystem som fungerer for både nordmenn, engelskmenn og svensker. I stedet for å hardkode valgene ”lenker”, ”hjem” og ”kontakt”, så lagrer du verdiene i en databasetabell, og lagrer samtidig også verdiene for andre språk, for eksempel ”links”, ”home” og ”contact”. I stedet for å bruke hardkodet JavaScript-kode, kan du nå generere JavaScript-koden på tjeneren, og dermed sende over en meny med engelsk innhold til brukere som har valgt engelsk som språk. Svensker får svensk tekst, nordmenn får norsk. PHP/ASP.NET + JavaScript er en flott kombinasjon!

Vi rekker ikke å ta turen innom Ajax – dette nye, spennende begrepet som kombinerer CSS, DOM, XHTML/HTML, JavaScript og XML til det vi kaller Asynkron JavaScript og XML (Ajax) og som åpner for virkelige brukervennlige løsninger. Du har sikkert sett mange Ajax-

baserte applikasjoner allerede. Hvis du vil lære mer om Ajax generelt og JavaScript spesielt, anbefales kurset "Web 2.0 med Ajax". Ajax er nemlig 80% JavaScript.

Lykke til med bruk av JavaScript på dine websider!