Liten JavaScript kurs rettet mot IT2

Johan Hake (johan.hake@gmail.com)

Aug 10, 2015

1 Forord

Dette er en ressurs for elever og lærere som ønsker å bruke javascript/HTML/CSS som programmeringsspråk i IT2-undervisningen. Hele nettstedet/boka har betastatus, og innhold vil fylles på og endres fortløpende. Dokumentet er basert på en WordPress versjon utarbeidet av Tom Jarle Christiansen. Han har også lagd de fleste av kodeeksemplene og youtube filmene. Et stort takk rettes derfor til Tom Jarle!

Dokumentet er bygget opp etter strukturen til kompetansemåla i faget, (se avsnitt: 3.3), og inneholder teori, oppgaver og videoer. Relevante kodeeksempler finnes på 'gist.github' og jsBin.

Dokumentet er lagd med hjelp doconce. Et stort takk rettes til visjonæren Hans Petter Langtangen som har utarbeidet dette verktøy.

Du står helt fritt til å bruke innholdet du finner i dette dokumentet slik du selv måtte ønske, så lenge du deler videre med vilkårene under.

JohanHake, August 2015

2 Teori

Den grunnleggende teorin i faget er delt opp i avsnitt som tilsvarer kompetansemålene i faget:

- Multimedieutvikling, se avsnitt 2.1
- Programmering, se avsnitt 2.2
- Planlegging og dokumentasjon, se avsnitt 2.3

2.1 Multimedieutvikling

Editor og filbehandling. Gode verktøy er helt avgjørende uansett hva slags jobb vi skal gjøre. Dette gjelder også når vi skal programmere. I denne boka skal vi benytte oss av to typer verktøy.

Nettleser: Program der koden kjøres og skjermbildet vises

Editor: Her skrives og editeres programkoden

Nettlesere. De fleste moderne nettlesere fungerer veldig bra som støtteverktøy til programmering. De har de funksjoner og verktøy som trengs for å jobbe med HTML-programmering på en effektiv og god måte. De mest brukte nettleserne i dag er Chrome, FireFox, Internet Explorer, Opera og Safari.

Editor. Det finnes veldig mange gode editorer for programmering. De mest avanserte er store programmeringsmijø, eller `IDE'er` som de også kalles. I dette kurset kommer vi langt med noe enklere editorer, som allikevel er veldig kraftige og funksjonsrike.

Til dette kurset anbefaler jeg at dere bruker `notepad++` eller `SublimeText` 2. Disse er enkle, men allikevel kraftige kodeeditorer der man enkelt og oversiktlig kan jobbe med flere filer samtidig. De har også god støtte for automatiske kodeforslag og fargelegging av kode slik at lesing går lettere.

Før vi lager vårt første html-side. Nå er vi straks klare til å sette i gang med det som er gøy, men før det må vi ha noen få ting på plass...

Før vi lager vårt første html-side må vi...

1. Lag en mappe som skal inneholde filene du lager. (Hvert program skal ha sin egen mappe).
2. Lag et nytt html-dokument som inneholder koden under. (Se under for startkoder).

Figure 1: Et lite ekesmpel på hvordan et enkelt HTML-dokument kan se ut.

Example: Kom igang med editorn.

Programkoden som du kan bruke i et HTML-dokument for å komme i gang.

```
<!doctype html>
<html lang="nb">
<head>
        <meta charset="UTF-8">
        <title>Mitt første HTML-dokument</title>
</head>
<body>
        <h1>Dette er en overskrift.</h1>
        <p>Dette er en paragraf, og denne teksten ser litt annerledes
          ut enn en overskrift</p>
</body>
</html>
```

HTML. **HyperText Markup Language** (HTML, hypertekstmarkeringsspråk) er et markeringsspråk for formatering av nettsider med 'hypertekst' og annen informasjon som kan vises i en nettleser. HTML benyttes til å strukturere informasjon - angi noe tekst som overskrifter, avsnitt, lister og så videre - og kan, i en viss grad, brukes til å beskrive utseende og semantikk i et dokument.

HTML ble opprinnelig definert i 1989 av TimBerners-Lee og RobertCaillau og videreutviklet av IETF og er nå en internasjonal standard (ISO/IEC 15445:2000). Siden har HTML-spesifikasjonene blitt opprettholdt av WorldWideWebConsortium (W3C)[1].

Programmering i HTML. Et HTML-dokument består av koder, eller "tagger" som beskriver hvordan vi vil at siden skal se ut, og i noen grad hva slags funksjonalitet den skal ha. I tabellen under finner du noen av de vanligste kodene som brukes[2].

[1] Hentet fra wikipediaartikkelen om HTML.

[2] For en utfyllende liste se http://www.w3schools.com.

Tag	Beskrivelse	Eksempel på bruk
<h1>...<h6>	Overskrift	Dette er en liten overskrift
	Avsnittstekst	Denne teksten blir samlet i et avsnitt.
	Fet skrift	Denne teksten blir fet
	Kursiv	Denne teksten er i kursiv
 	Linjeskift	Tekst på en linje Tekst på linja under
<hr>	Horisontal linje	<hr>
<a>	Hyperlenke	 Fagnettsted
<a>	Epostlenke	 Send e-post
<img>	Sette inn bilde	<img src="mittBilde.jpg">
og	Punktliste	 - Punkt 1 - Punkt 2
og	Nummerert liste	 - Punkt 1 - Punkt 2
<table>, <tr> og <td>	Tabell	<table> <tr><td>Celle1</td><td>Celle2</tr> <tr><td>Celle1</td><td>Celle2</tr> </table>

Fra tabellen ser vi at noen tagger trenger en lukketag, som sier nåt taggen slutter, f eks: , </tr> og </td>. Hvis ikke en lukketag følger den tilsvarende åpningstaggen får vi et syntaksfeil og siden kan oftest ikke vises. Noen tagger trenger ikke en lukketag, f eks: <hr> og
.

HTML-koden settes sammen i et HTML-dokument som igjen kan vises i en nettleser.

```
<!doctype html>
<html lang="nb">
<head>
        <meta charset="UTF-8">
        <title>Mitt første HTML-dokument</title>
</head>
<body>
        <h1>Dette er en overskrift.</h1>
        <p>Dette er en paragraf, og denne teksten ser litt annerledes
          ut enn en overskrift</p>
</body>
</html>
```

Hvert html dokument må ha noen standard tagger som <!doctype> og <html>. doctype taggen sier til en netleser at det er et html dokument og html taggen sier hvor selve html koden kommer. Vi ser fra eksempelet at <html> taggen har en så kallt attributt: lang="no", som er tileggsinformasjon til taggen, se nedan. På rad 3-6

defineres `<head>` `taggen`, som bestemmer egenskaper til hele html siden, for eksempel titel og tegnsett (encoding). En html dokument bør også ha en `<body>` tag. Denne `taggen` inneholder det som skall vises på siden.

Example: Test tagger.

Test ulike tagger gjennom å laste in filen i en editor og en nettleser

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Tagtesting!</title>
</head>
<body>
    <h1>Test tagger her!</h1>
    <p>Prøv deg selv med egen kode ved å skrive i HTML-fanen.</p>
</body>
</html>
```

Attributter. Attributter brukes for å gi ekstra informasjon til hva taggen skal gjøre. En attributt plasseres inne i taggen, og kan for eksempel inneholde informasjon om bakgrunnsfarge, plassering eller identifikatorer som skal gjelde for den aktuelle taggen. Det er fullt mulig å ha flere attributter i samme tag.

Attributtnavn	Egenskap	Eksempel
`align`	Plassering av teksten	`<h1 align="right">Denne overskriften vil stå på høyre side</h1>`
`bgcolor`	Bakgrunnsfarge	`<h1 bgcolor="green">Denne overskriften vil ha grønn bakgrunn</h1>`
`height og width`	Høyde og bredde på bilder, tabeller og celler	`<table width="300px">`

Attributter er en viktig del av et HTML-dokument, og gjør at vi kan *finjustere* taggene til å vise innhold slik vi ønsker. For mer avansert design og utseende må vi benytte oss av CSS.

Her brukes attributtene src og width for å gi egenskaper til bilde-taggen

```
<body>
  <h1>Min timeplan</h1>
  <img src="bilder/klasserom.jpg" width="400px">
</body>
```

input. Veldig ofte ønsker vi at brukeren skal legge inn informasjon slik at den kan lagres eller bearbeides. For å få til dette kan vi bruke `<input>`. Ved å bruke denne taggen sammen med ulike attributter kan vi lage elementer som tekstbokser, radioknapper, draknapper, databokser osv. `<input>` brukes ofte sammen med `<form>` som samler alle input-tagger i et skjema. Selv om `<input>` kun er en enkelt tag, så kan man få den til å ha mange former ved å bruke attributten `type`.

Eksempel på input-tag

```
<form>
Fornavn: <input type="text"><br>
Etternavn: <input type="text">
</form>
```

Det finnes mange ulike typer inputfelt. I tabellen under finner du eksempel på flere.

Type	Beskrivelse
button	Vanlig trykknapp
color	Fargevelger
date	Datovelger
datetime	Dato- og tidsvelger
email	Tekstfelt for epost
month	Månedsvelger
password	Tekstfelt for passord
radio	Radioknapper
range	Glideknapp
reset	Nulstiller alt som er inne i samme form
search	Søkefelt
submit	Sender all data som er i samme form
tel	Felt for telefonnummer
text	Tekstfelt
time	Klokkeslett
url	Felt for nettadresser
week	Ukevelger
checkbox	Avkryssingsbokser
number	Tallvelger

Example: Test input.

Test ulike input-typer gjennom å laste in filen i en editor og en nettleser

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JS Bin</title>
</head>
<body>
    <h1> Her kan du teste ulike input-typer</h1><br>
    <p>Prøv for eksempel: "color", "checkbox" eller "number".
    <input type="text">
</html>
```

taggene video og audio. `Video` og `audio`-taggene er brukes når man skal spille av film og lyd. Flere formater støttes, og man har mulighet for volum og avspillingskontroll.

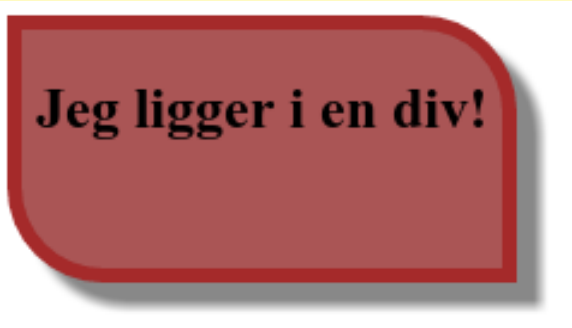
Eksempel på audio og video-tagger med ulike attributter

```
<audio controls>
    <source src="lyd.ogg">
</audio>
<video width="320" height="240" controls>
  <source src="film.ogg">
</video>
```

div. `<div>` er en fin måte å lage et avgrenset område på nettsiden. `<div>` i seg selv er kun et tomt skall, og vi er nødt til å fylle på med innhold mellom start- og slut-taggen. Ved å bruke attributter på div-taggen kan vi få den til å se ut på mange forskjellige måter.

Example: Lek med div.

Her kan du endre på div-style attributten selv gjennom å laste in filen i en editor og en nettleser




```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Leke med div</title>
</head>
<body>
        <div  style="text-align: center;
            width: 200px;
            height: 100px;
            background-color: #aa5555;
            border-color: brown;
            border-width: 6px;
            border-style: solid;
            border-radius: 0px 40px;
            box-shadow: 10px 10px 5px #888888;">
      <h2>Jeg ligger i en div!</h2></div>
</body>
</html>
```

Det finnes så mange html-tagger og ulike attributter at de færreste husker alle selv. Det er derfor viktig å lære seg å slå opp. W3schools.com er et nettsted der du selv kan finne tagger, og se eksempler på hvordan de brukes.

DOM treet. Når en webside lastes skaper nettleseren en Document Object Model (DOM) av siden. HTML DOM modellen skapes som et tre av objekter, se figur 2. DOM er en W3C standard, som definerer metoder for å få tilgang på og endre et HTML (eller XML) dokument.

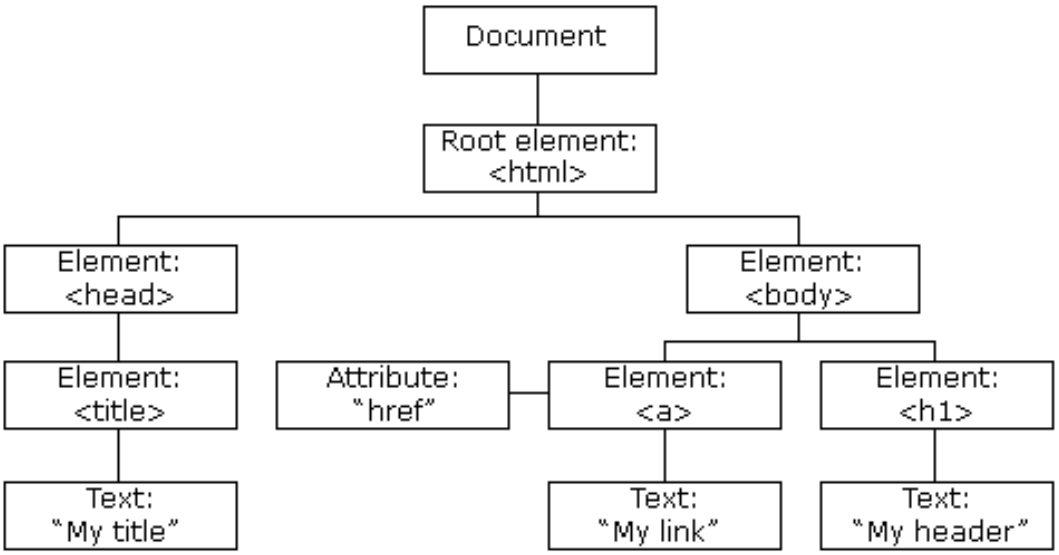


Figure 2: Et HTML DOM tre av objekter (hentet fra 'www.w3schools.com').

CSS.

2.2 Programmering

2.3 Planlegging og dokumentasjon

3 Læreplan i informasjonsteknologi

Dette avsnittet er hentet fra udir.

3.1 Formål

Informasjonsteknologien har hatt stor betydning for samfunnsutviklingen de siste tiårene. Teknologien har i løpet av kort tid endret kommunikasjonsmønsteret i samfunnet og skapt nye arbeidsplasser og lærings- og forskningsarenaer. Samfunnet har behov for mennesker som kan forstå, benytte og videreutvikle informasjonsteknologien, men samfunnet trenger også mennesker med en bevisst og kritisk holdning til hva teknologien gjør med mennesker og samfunn. Informasjonsteknologien gir muligheter for å lage helt nye produkter og tjenester gjennom kreativitet og samarbeid over faggrenser, og bidrar dermed til teknologisk innovasjon.

Programfaget skal gi trening i kreativ tenkning og problemløsning og i å formulere presise beskrivelser og finne generelle mønstre. Programfaget skal bidra til å gi innsikt i hvordan informasjon i form av tall, tekster, bilder, grafikk, film, lyd og animasjoner kan struktureres og behandles automatisk som data, og hvilke krav det setter til datamaskiner og annet digitalt utstyr. Gjennom programfaget skal den enkelte få erfaring med bruk av moderne teknologi og relevante utviklingsverktøy, og hvordan sammensmelting av data-, lyd- og bildeteknologi kan gi rom for skapende bruk av teknologien.

Opplæringen legger vekt på å konstruere IT-løsninger, og informasjonsteknologi er derfor på mange måter et praktisk fag. Det skal legges til rette for kommunikasjon og samarbeid i programfaget. Programfaget informasjonsteknologi er et realfag, men det har også sterke koblinger til mediefag, samfunnsfag, økonomi, språkfag og formgivingsfag. Programfaget kan derfor gi et godt grunnlag for studier innen ulike fagområder og for videreutvikling av kompetanse i yrkeslivet.

3.2 Grunnleggende ferdigheter

Grunnleggende ferdigheter er integrert i kompetansemålene der hvor de bidrar til utvikling av og er en del av fagkompetansen. I informasjonsteknologi forstås grunnleggende ferdigheter slik:

Å kunne uttrykke seg muntlig og skriftlig i informasjonsteknologi innebærer å planlegge og beskrive IT-løsninger, og utarbeide brukerveiledninger og dokumentasjon. Videre vil det si å formulere presise instruksjoner for datamaskiner i et programmeringsspråk. Det betyr også å uttrykke seg på en klar og presis måte.

Å kunne lese i informasjonsteknologi innerærer å tolke beskrivelser, brukerveiledninger, diagrammer, modeller, symboler og programkode på en presis måte. Videre betyr det å forstå fagspesifikke tekster.

Å kunne regne i informasjonsteknologi innebærer å gjøre enkle utregninger eller uttrykke formler i et programmeringsspråk. Videre vil det si å bruke enkel matematisk logikk for å uttrykke en betingelse på en presis måte.

Å kunne bruke digitale verktøy utgjør en grunnstamme i informasjonsteknologi. Det innebærer å bruke IT-løsninger på en effektiv måte og bruke digitale verktøy både i planleggings- og dokumentasjonsprosesser.

3.3 Kompetansemål IT2

Planlegging og dokumentasjon. Hovedområdet handler om planlegging av IT-løsninger, og utvikling av disse etter gitte spesifikasjoner for å oppfylle brukernes behov. Videre dreier det seg om dokumentasjon og vurdering av IT-løsninger. Hovedområdet omfatter også utforming, dokumentasjon og vurdering av løsninger i forhold til retningslinjer for brukergrensesnitt.

Mål for opplæringen er at eleven skal kunne:

1. spesifisere og begrunne funksjonelle krav til planlagte IT-løsninger
2. velge og bruke relevante teknikker og verktøy for planlegging og utvikling av IT-løsninger
3. lage brukerveiledninger for IT-løsninger
4. gjøre rede for hvordan IT-løsninger utvikles i samarbeid mellom personer, og hvilke krav det setter til planleggings- og utviklingsprosessen
5. forklare hensikten med teknisk dokumentasjon og lage slik dokumentasjon for IT-løsninger, med spesiell vekt på å dokumentere grensesnitt mellom ulike delsystemer

Programmering. Hovedområdet handler om hvordan formelle språk kan brukes til å formulere strukturer og sette sammen instruksjoner som kan utføres av en datamaskin. Sentralt i hovedområdet er eksperimentering og problemløsning. I tillegg dreier det seg om objektorientering.

Mål for opplæringen er at eleven skal kunne:

1. lese og bruke dokumentasjon og kode
2. definere variabler og velge hensiktsmessige datatyper
3. tilordne uttrykk til variabler
4. programmere med enkle og indekserte variabler eller andre kolleksjoner av variabler
5. programmere med valg og gjentakelser
6. lage egne og bruke egne og andres funksjoner eller metoder med parametere

7. programmere funksjoner eller metoder som blir aktivisert av hendelser
8. utvikle og sette sammen delprogrammer
9. teste og finne feil i programmer ved å bruke vanlige teknikker
10. gjøre rede for hensikten med objektorientert utvikling og begrepene klasse, objekt og arv

Multimedieutvikling. Hovedområdet handler om utforming, strukturering, implementering og vurdering av multimedieapplikasjoner med tall, tekst, lyd, bilde, video og animasjoner. I tillegg omfatter hovedområdet både applikasjoner for lokal bruk og publisering over Internett.

Mål for opplæringen er at elevene skal kunne:

1. planlegge og utvikle multimedieapplikasjoner ved å kombinere egne og andres multimedieelementer av typene tekst, bilde, lyd, video og animasjoner
2. bruke programmeringsspråk i multimedieapplikasjoner
3. vurdere og bruke relevante filformater for tekst, bilde, lyd, video og animasjoner
4. vurdere multimedieprodukter med hensyn til brukergrensesnitt og funksjonalitet