

Internetværk og Web-programmering

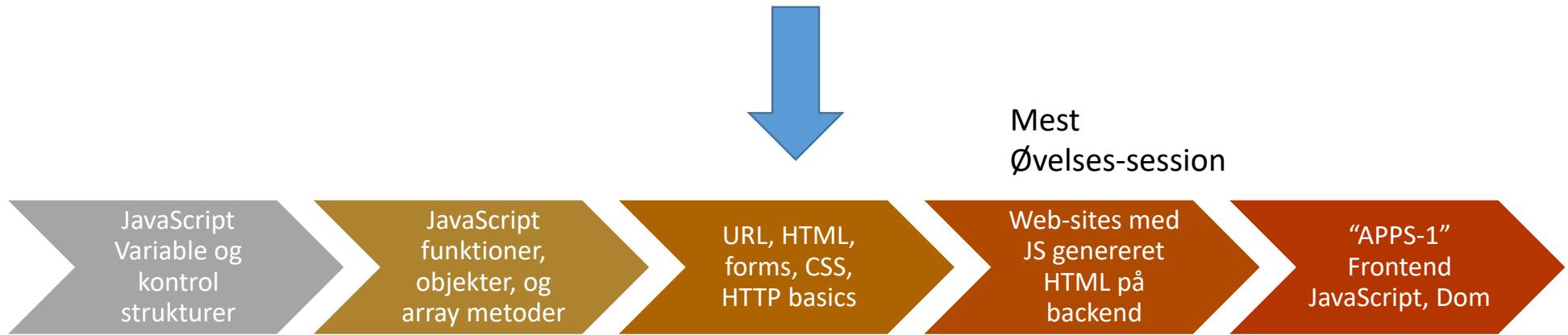
HTML, HTTP, Forms, CSS

Forelæsning 3
Brian Nielsen

Distributed, Embedded, Intelligent Systems



Lektions-status



URLs

Uniform Resource Locators

- En henvisning til en tilgængelig ressource aka **web-adresse**
- Ressource: den “ting” man ønsker at få/give adgang; fx en web-side, video, billede, applikation, fil, IoT Device, ...
- Angiver hvor ressourcen befinner sig, og hvordan den tilgås



Protokol (mere korrekt ”scheme”): Angiver den service (HTTP, FTP, file, mailto, tel, IRC,...) browseren skal tilgå (ofte svarende til en applikationsniveau protokol)

Domænet: DNS navn eller IP adresse, samt evt. port nummer "130.225.63.3:8080"

- Default: 443 (HTTPS) 80 (HTTP)

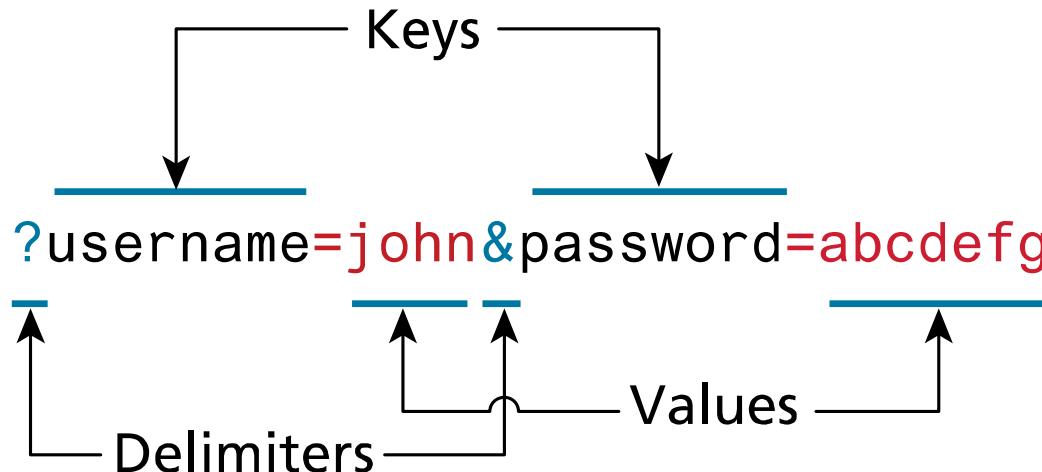
Sti-navn: Navn på ressource indenfor det pågældende domæne: oversættes sommetider til et filnavn

Søgestreng: en række ”key-value” par som server program kan bruge til at finde den ønskede (del af) ressourcen: fungerer som parameterliste til scripts

Fragment: en måde at bede om en portion af et dokument.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Identifying_resources_on_the_Web

Uniform Resource Locators



- URL kan kun sendes på nettet som ASCII: Derfor overføres de "URL-encodet"
 - Skjules for det meste af nyere browsere
- Special tegn og reserverede tegn, fx ?&+()[]#=, skal "escapes"
- Escape karakter i URL: %
- URL-encoding: : https://www.w3schools.com/tags/ref_urlencode.ASP
- Fx. æbelø? -> %C3%A6bel%C3%B8%3F
- [DF 11.9] beskriver URL og SearchParams" APIs

Markup og HTML

Hypertext & Markup-sprog

- **Hypertext:** Samle tekst information og forbinde disse via henvisninger, så informationen kan læses i bruger-defineret rækkefølge.
 - Gammel ide: '45,
 - '65: termen "hypertext" brugt af Ted Nelson i 1965 [wikipedia]
 - '86 Apples Hypercard system
 - Hypermedia (linked multi-medier)
 - '89 www draft: <https://home.cern/science/computing/birth-web/short-history-web>
- **"Markup"** er markeringer, tilføjelser, og annoteringer, der tilføjer ekstra information om et (tekst) dokument.
- **Markup sprog:** det computer sprog (syntax og semantik) der anvendes.
 - LaTeX tilføjer struktur og typesetting information.
 - **HTML** (Hyper Text Markup Language) udviklet af Tim Berners-lee 1991 til www hyper-tekst documenter
 - YAML: yet-another-markup-language
 - MD: Markdown
 -

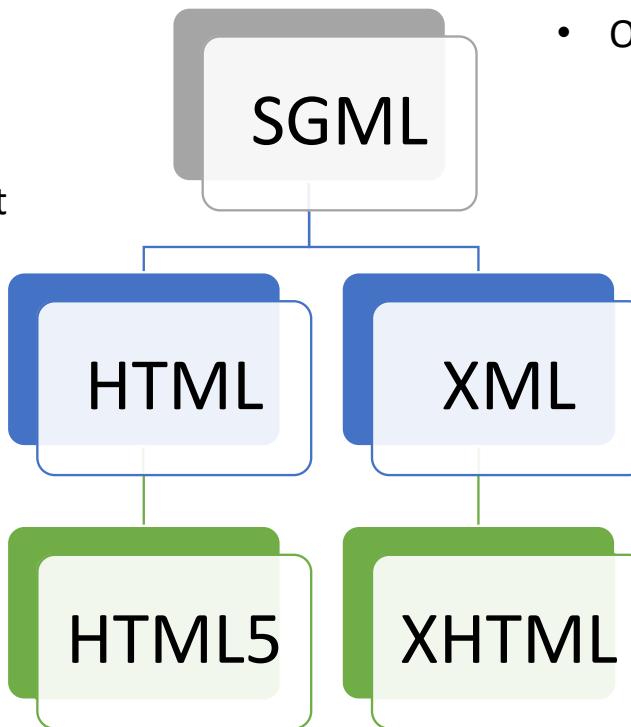
Markup-sprog

- **HTML** (Hyper Text Markup Language) udviklet af Tim Berners-lee 1991, inspireret af SGML, til www hyper-tekst documenter
- Til visning af hypertext documenter
- HTML5 (2012) drives af Web Hypertext Application Technology Working Group ([WHATWG](#))

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a heading (first one) </h1>
<p> The paragraph </p>

</body>
</html>
```



- **Standard Generalized Markup Language** (1986; [ISO](#) 8879) er en standard til at definere markup-sprog til elektroniske dokumenter.
- Omfangsrigt og komplext

- **eXtensible Markup Language** (W3C, 1998) **delmænge** af SGML til brug i WWW, som var nemmere at lave værktøjer til.
- Brugerne definerer selv deres tilladte elementer, "tags", nesting, etc.
- Mål: beskrive hvad data "består af"
- En instans af et XML dokument:

```
<><students>
  <><student id="100026">
    <><name>Joe Average</name>
    <><age>21</age>
    <><major>Biology</major>
  <><results>
    <><result course="Math 101" grade="C-"/>
    <><result course="Biology 101" grade="C+"/>
    <><result course="Statistics 101" grade="D"/>
  <></results>
<></student>
  ><><student id="100078">...</student>
<></students>
```

Semantisk Markup

- Fokus på at definer struktur af dokumentet, ikke det visuelle
- Har en række fordele
 - Nemmere at **vedligehold**: farvelade og layout kan gives separat, og fælles for alle elementer, fx overskrifter.
 - **Performance**: Mindre dokumenter, der ikke indeholder en masse visuelle instruktioner.
 - Markering af strukturen kan **øge tilgængelighed** for svagtseende “Accessibility” (<http://www.w3.org/WAI>).
 - **Søgemaskine optimering**: Ord som er semantisk fremhævede, eller som fremtræder i titlen og overskrifter gives højere prioritering rangering af søgeresultater

HTML5 Syntax: Elementer og Attributter

- Et **HTML document** består af tekstuelt indhold og **HTML elementer**
- **HTML element** består af
 - **element navn** er omsluttet af <> paranteser (også kaldet **tag**)
 - **attributer**
 - **Indhold** mellem start og slut tag.
- Fx **anchor element** (laver hyperlink)



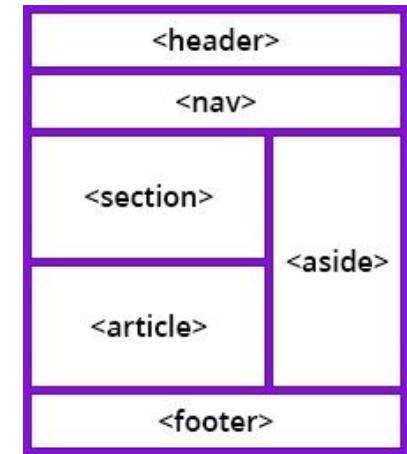
- “**Tom element**” har ikke noget tekst indhold og intet slut tag.

```

```

HTML Elementer

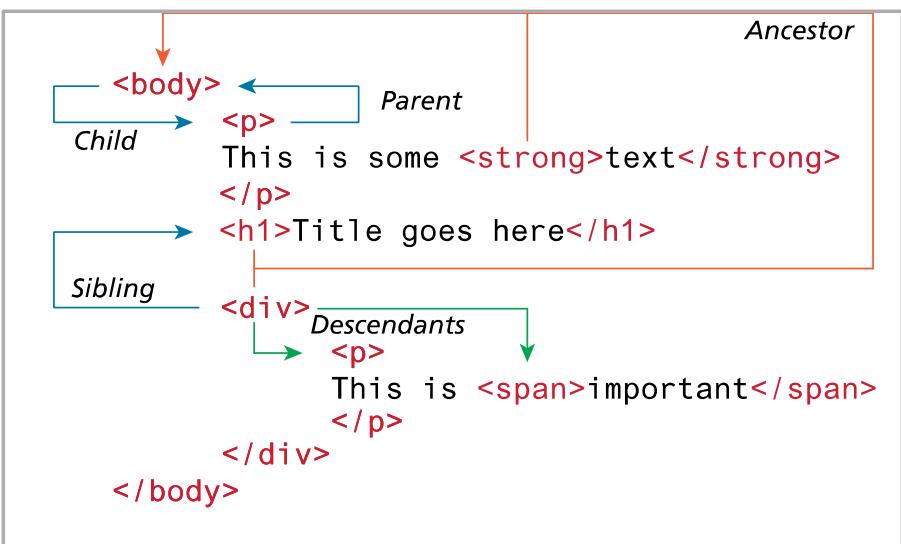
- Ca. 110 forskellige (32 nye i HTML5)
 - Strukturelle og sektionering: Opdeling af dokument i separate logisk sammenhængende områder
 - Opdeling i **tekst-blokke**: Opdeling af sektion i mindre, samhørende dele; laver linieskift
 - **Inline tekst**: Definerer betydning/stil af et ord indenfor en tekst-linie; uden at lave linieskift.
 - Billeder, 2D- og 3D grafik, multi-medier
 - Tabeller
 - Bruger-input, formularer ("Forms"), interaktive dialoger og menuer
 - Scripting og templates
 - Meta-data: Information om dokumentet (fx karaktersæt, forfatter., direktiver til søgemaskiner ...)
- Se fx oversigt på:
 - <https://html.spec.whatwg.org/multipage/>
 - <https://developer.mozilla.org/en-US/docs/Web/HTML/Element>
 - <https://www.creativebloq.com/advice/html-tags>
 - <https://www.atnyla.com/tutorial/introduction-to-html5/2/238>



God brug er **vigtig** for
Muligheder for at lave egen layout og styling
Brugbarhed for svagtseende
Optimering for søgemaskine

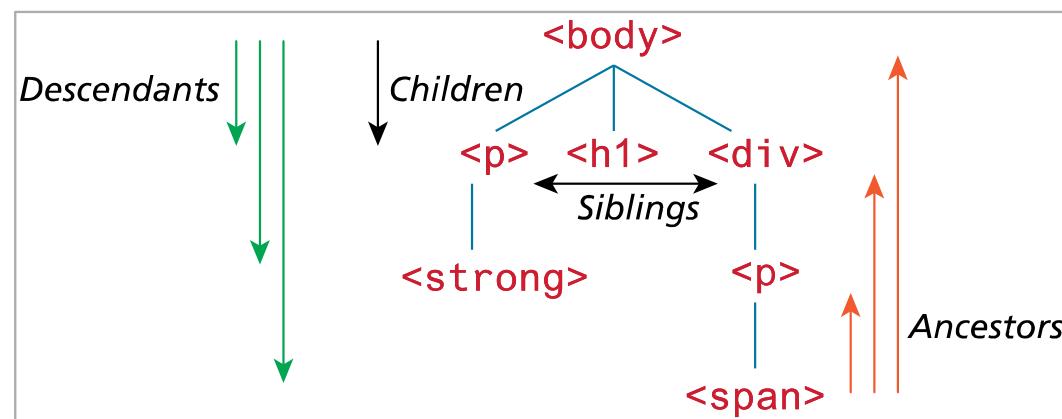
HTML Syntax: Nesting

Som parenteser, der skal
balance



NB Vigtigt:

Elementernes nesting kan vises som et træ:



Relationer som alm. familie træ:

- Elementer på samme niveau: **Søskende**
 - Elementer på underliggende niveauer: **etterfølgere**
 - Elementer på overliggende niveauer: **forfædre**
 - Elementer på umiddelbart overliggende niveau: **forældre**
 - Elementer på umiddelbart underliggende niveau: **børn**

class og id attributter

- Alle html elementer kan gives en id attribut
 - Fx <p **id="id1"** > Dette er en paragraf</p>
 - Tekst strenge uden whitespace;
 - Id-attribut skal være unikt i dokumentet
 - Muligt at udpege det givne id som et *"anchor"*
 - JS/CSS kan udpege elementet ud fra id.
- Alle html elementer kan grupperes i bruger-definerede klasser
 - Fx <p **class="note"** > Dette er en paragraph </p>
 - Gør et muligt for JavaScript / CSS at udpege alle elementerne med det givne klasse-navn
 - Kan samtidigt have id.
- Se fx:
 - https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/class
 - https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/id

IWP Demo Side is a crude HTML page made for demo purposes
Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sød Panda som browser bambus! © National Geographic

According to [Goodreads](#), [Turing Award Winner Edsger Dijkstra](#), has stated that "If debugging is the process of removing software bugs, then programming must be the process of putting them in."

Du skal dog først kende til [HTML](#).

[Forfattet af Brian Nielsen](#)

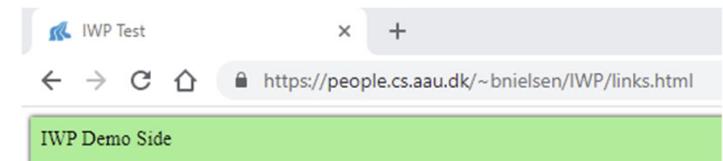
Resultat med browsers default stylesheet

```
<body>
  <header> IWP Demo Side is a crude HTML page made for demo purposes </header>
  <nav>
    Indholdsfortegnelse:
    <ul>
      <li><a href="#htmlafsnit"> Introduktion </a>
      <ul>
        <li><a href="#htmlafsnit"> HTML </a></li>
        <li><a href="#htmlafsnit"> JavaScript. </a> </li>
      </ul></li>
      <li> to do </li>
    </ul>
  </nav>
  <aside>Ingen bemærkninger</aside>
  <main>
    <section id="introsection">
      <h1> Introduktion til Web-Programmering</h1>
      <em> Centrale</em> elementer i web-teknologi er <a href="#htmlafsnit"> HTML </a> og <a href="#htmlafsnit"> JavaScript. </a>
      <p>la bla bla.</p>
      <p> Klient-side programmering kommer i <a href="lecture5.html">næste lektion. </a> </p>
      Her er en god <a href="https://developer.mozilla.org/" title="God Web Reference Site">reference til web-teknologi </a>
      </section>
      <section id="htmlafsnit">
        <h2> HTML </h2>
        Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc.
        Etc.
        <h3>HTML BODY</h3> indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.
        </section>
        <!-- HER ER EN KOMMENTAR -->
        <section id="jsafsnit">
          <h2>JavaScript</h2>
          JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.
          <figure>
            
            <figcaption>Fig. 1 - Sød Panda som browser bambus! &#169; National Geographic </figcaption>
          </figure>
          According to <cite>Goodreads</cite>, <a href="https://da.wikipedia.org/wiki/Turing-prisen">Turing Award Winner</a> <a href="https://en.wikipedia.org/wiki/Edsger_W._Dijkstra"> Edsger Dijkstra:</a> has stated that
          <q cite="https://www.goodreads.com/author/quotes/1013817. Edsger_W._Dijkstra" >
            If debugging is the process of removing software bugs,
            then programming must be the process of putting them in.
          </q>
          <p> Du skal dog først kende til <a href="#htmlafsnit"> HTML </a>. </p>
        </section>
      </main>
      <footer>
        <a href="https://www.cs.aau.dk/~bnielsen/" rel="author">Forfattet af Brian Nielsen</a>
      </footer>
    </body>
```

<http://people.cs.aau.dk/~bnielsen/IWP/links.html>

HTML header og body

```
<!DOCTYPE html>
<html lang="da">
  <head>
    <title>IWP Test</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <header> IWP Demo Side</header>
  </body>
</html>
```



1. Doctype fortæller browser at det kommende tekster et et HTML5 dokument
2. <html> er rod-elementet, som markerer start og slut.
 - Det er god praksis at sætte sprog attributtet afhængigt skærmblæsere.
 - Efterfølges af ét <head> og ét <body> element
3. <head> indeholder info om dokumentet. Kan indeholde mange ting! Typisk mindst:
 - Sætter titlen på siden; vises øverst i browseren vindue/tab
 - Definerer det karaktersæt der anvendes i dokumentet. UTF-8 tillader de fleste naturlige sprogs tegn – anbefalet.
 - Brugte eksterne stilarter
 - Brugte eksterne javascripts
4. <body>indeholder selve indholdet som skal vises på siden
 - <header> er et alm. element indeholder introducerende indhold

Meta-data: data som er information om andet data

https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started#Anatomy_of_an_HTML_document

IWP Demo Side is a crude HTML page made for demo purposes
Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to *Goodreads*, [Turing Award Winner Edsger Dijkstra](#), has stated that “If debugging is the process of removing software bugs, then programming must be the process of putting them in.”

Du skal dog først kende til [HTML](#).

Forfattet af Brian Nielsen

HTML Ex: struktur elementer

```
<body>
  <header> IWP Demo Side is a crude HTML page
    | made for demo purposes </header>
  <nav> ...
  </nav>
  <aside>Ingen bemærkninger</aside>
  <main>
    <section id="introsection"> ...
    </section>

    <section id="htmlafsnit"> ...
    </section>
    <!-- HER ER EN KOMMENTAR -->
    <section id="jsafsnit">...
    </section>
  </main>
  <footer>...
  </footer>
</body>
```

IWP Demo Side is a crude HTML page made for demo purposes
Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to *Goodreads*, [Turing Award Winner Edsger Dijkstra](#), has stated that “If debugging is the process of removing software bugs, then programming must be the process of putting them in.”

Du skal dog først kende til [HTML](#).

[Forfattet af Brian Nielsen](#)

HTML Ex: headings og paragraffer

```
<section id="introsection">
  <h1> Introduktion til Web-Programmering</h1>
  <em> Centrale</em> elementer i web-teknologi er ...
  <p>la bla bla.</p>
  <p> Klient-side programmering kommer i
      <a href="lecture5.html"> næste lektion. </a> </p>
</section>
```

<http://people.cs.aau.dk/~bnielsen/IWP/links.html>

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- [to do](#)

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to Goodreads, [Turing Award Winner Edsger Dijkstra](#), has stated that " If debugging is the process of removing software bugs, then programming must be the process of putting them in. "

Du skal dog først kende til [HTML](#).

Forfattet af Brian Nielsen

HTML Ex: hyperlinks, fragmenter og relative URLs

<https://homes.cs.aau.dk/~bnielsen/IWP/links.html>

```
<section id="introsection">
  <h1> Introduktion til Web-Programmering</h1>
  <em> Centrale</em> elementer i web-teknologi er <a href="#htmlafsnit"> HTML
  </a> og <a href="#jsafsnit"> JavaScript. </a>
  <p>la bla bla.</p>
  <p> Klient-side programmering kommer i <a href="lecture5.html">næste
  lektion. </a> </p>
  Her er en god <a href="https://developer.mozilla.org" title="God Web
  Reference Site">reference til web-teknologi </a>
</section>

<section id="htmlafsnit">
  <h2> HTML </h2>
  Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-
  browser. Etc. Etc. Etc.
  <h3>HTML BODY</h3> indeholder vigtig information om dokumentet, som
  modtageren skal bruge for at forstå det.
</section>
```

Relative URLs bliver fortolket relativt til omsluttende doc:

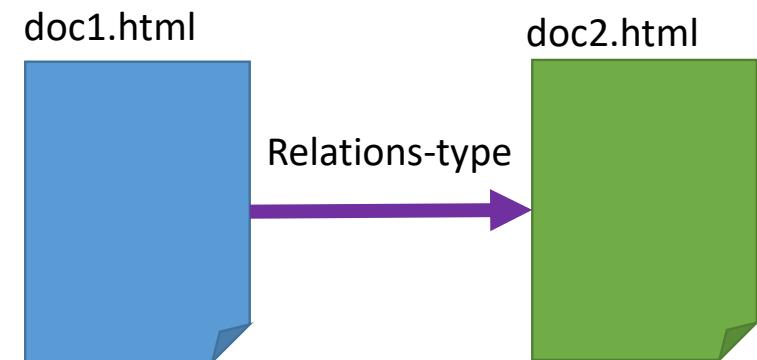
- <https://homes.cs.aau.dk/~bnielsen/IWP/lecture5.html>
- <https://homes.cs.aau.dk/~bnielsen/IWP/links.html#htmlafsnit>

Der er flere attributter på anchor elementet: fx

- title: angiver formålet med linket (vises som et tool-tip)
- "download"
- Se i reference manualen: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a>

Lidt mere om hypertext "relationer"

- <link> eller rel attribut
 - rel="stylesheet"
 - Udpeger stil-regler
 - rel="author"
 - rel="alternate"
 - Fx en mobil udgave
 - rel="hreflang"
 - Forskellige sproglige versioner
 - rel="prev", rel="next"
 - Kapitlerne i en bog
 - ...
- Bruges af programmer som behandler hypertext dokumenter, fx søgemaskiner.
- Information i meta-elementet og links/relationer giver mulighed for søgemaskine optimering ("SOE")



```
<a href="https://www.cs.aau.dk/~bnielsen" rel="author">  
Forfattet af Brian Nielsen</a>  
  
<link rel="alternate" hreflang="es"  
      href="https://www.example.es" >
```

IWP Demo Side is a crude HTML page made for demo purposes
Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to *Goodreads*, [Turing Award Winner Edsger Dijkstra](#), has stated that “If debugging is the process of removing software bugs, then programming must be the process of putting them in.”

Du skal dog først kende til [HTML](#).

Forfattet af Brian Nielsen

HTML Ex: Figurer, billeder, special tegn

```
<figure>
  
  <figcaption>Fig. 1 -
    Sød Panda som browser bambus!    &#169; National Geographic
  </figcaption>
</figure>
```

Figurer har også en titel / caption

Billeder bør altid have en ”alt” beskrivelse afh. skærmlæsere

- Reserverede tegn "<>& kan skrives ”Entity”: &streng; fx "
- [1000 vis af andre special tegn](#)

IWP Demo Side is a crude HTML page made for demo purposes
Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Ingen bemærkninger

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to [Goodreads](#), [Turing Award Winner Edsger Dijkstra](#), has stated that “If debugging is the process of removing software bugs, then programming must be the process of putting them in.”

Du skal dog først kende til [HTML](#).

Forfattet af Brian Nielsen

HTML Ex: citationer

According to **<cite>**Goodreads**</cite>**,
[a href="https://da.wikipedia.org/wiki/Turing-prisen"](https://da.wikipedia.org/wiki/Turing-prisen)
Turing Award Winner[a href="https://en.wikipedia.org/wiki/Edsger_W._Dijkstra"](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) Edsger Dijkstra:[a href="https://en.wikipedia.org/wiki/Edsger_W._Dijkstra"](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) has stated that
<q cite="https://www.goodreads.com/author/quotes/1013817.Edsger_W_Dijkstra">
If debugging is the process of removing software bugs,
then programming must be the process of putting them in.
</q>

- **<cite>** angiver artiklen, der er citeret
- **<q>** er citatet; dets cite attribut udpeger citatets kilde

HTML Tabel Ex

Scoreboard

Yatzy Scores		
Peter		
Round Name	Dice	Score
1s		0
2s		0
3s		0
4s		0

- Tabel har en **caption**, **head**, **body**
- Heading kolonner angives med **<th>**
- Rækker angives med **<tr>**
- Data kolonner angives med **<td>**
- **class** attribut bruges af stylesheet til at højre/venstre stille tekst, og give grøn baggrund til special runder
- Terninger er bare separate billeder
- Terninger samles i en "container" enhed: **span** (alt. **div**)

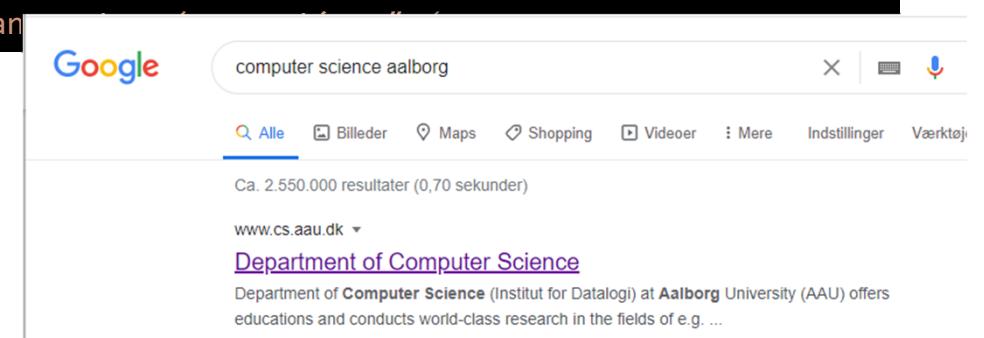
```
<table id="scoretable">
  <caption> Yatzy Scores </caption>
  <thead>
    <tr>
      <th colspan="3"> Peter </th>
    </tr>
    <tr>
      <th>Round Name </th>
      <th> Dice </th>
      <th>Score</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td class="left-text"> 1s </td>
      <td> <span>
          
          
          ...
        </span>
      </td>
      <td class="right-text"> 0 </td>
    </tr>
    ...
    <tr class="row-fill"> <td>Sum</td>...</tr>
    ...
  </tbody>
</table>
```

Mere om HTML Head

```
<head>
  <meta charset="UTF-8">
  <title>IWP Test</title>
  <link rel="stylesheet" href="style.css">
  <link rel="author" href="http://www.cs.aau.dk/~bnielsen">
  <link rel="alternate" hreflang="es" href="https://www.example.es" >
  <meta name="copyright" content="Brian Nielsen" >
  <meta name="description" content="EN lille IWP Demo, som viser nogle interessante aspekter af HTML">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
...

```

```
<meta name="description" content="Department of Computer Science (Institut for Datalogi) at Aalborg University (AAU) offers educations and conducts world-class research in the fields of e.g. embedded software systems, data-intensive systems, artificial intelligence and human
```



Validering af HTML dokumenter og CSS

- Hensigten bag HTML elementerne er beskrevet i
 - <https://html.spec.whatwg.org/>
 - (The Web Hypertext Application Technology Working Group (WHATWG) is a community of people interested in evolving the web through standards and tests.)
- Der findes online validatorer, som kan checke for syntax og enkelte semantiske fejl i HTML dokumenter og CSS
 - <https://validator.w3.org/>

CSS Ex. Layout og farvelader

IWP Demo Side is a crude HTML page made for demo purposes

Indholdsfortegnelse:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- [to do](#)

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).

la bla bla.

Klient-side programmering kommer i [næste lektion](#).

Her er en god [reference til web-teknologi](#)

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sod Panda som browser bambus! © National Geographic

According to [Goodreads](#), [Turing Award Winner Edsger Dijkstra](#) has stated that " If debugging is the process of removing software bugs, then programming must be the process of putting them in. [

https://www.goodreads.com/author/quotes/1013817.Edsger_W_Dijkstra]

Du skal dog først kende til [HTML](#).

Ingen
bemærkninger

```
body {  
    display: grid;  
    grid-template-areas:  
        "h h h"  
        "n m a"  
        "f f f";  
    grid-template-rows: 70px 1fr 60px;  
    grid-template-columns: 20% 1fr 15%;  
    grid-row-gap: 10px;  
    grid-column-gap: 10px;  
    height: 100vh;  
    margin: 0;  
}  
/*general coloring and layout */  
header, footer, main, nav, aside {  
    padding: 0.5em;  
    background:lightgray;  
    margin: 0.5em;  
    color: black; box-shadow: 0 0 0.25em black;  
}  
header { grid-area: h; }  
footer { grid-area: f; }  
main { grid-area: m; }  
nav { grid-area: n; }  
aside { grid-area: a; }  
header{  
    background-color: lightgreen;  
}  
  
section{  
    margin:1em;  
    border: thin dashed; padding:1em;  
    background: rgb(190, 190, 190);  
}  
#introsection{ background: rgb(248, 229, 55);}  
main{  
    /* try to uncomment this */  
    /*display: flex;*/  
}
```

HTML Input og Forms

HTML Formularer

- HTML5 tilbyder et større antal elementer typer til bruger input:
 - <button> <datalist> <fieldset> <input> <form> <label> <legend> <meter> <output> <optgroup> <progress> <textarea> <select>
 - <input> undertyper: button, checkbox, radio, range, text, number, submit ,...
- Kan bruges enkeltvist til input til JS program
- Kan samles i en formular <form> </form> med en “submit” knap
 - Kan have en “action” URL og en “http metode” attribut, som normal får browser til at sende formular data i et http request, og som respons forventer en ny side med resultatet
- Alternativt, kan formularen kan også behandles af JS:
 - Validering og fejl-retning inden formularen indsendes
 - Kun JS behandler dataene

The diagram shows a form titled "Personal information:" enclosed in a border. It contains three text input fields and one button:

- Name: Mickey
- Height (cm): (empty input field)
- Weight (kg): (empty input field)
- Record: (button)

Three purple arrows point from the text "Name", "Weight (kg)", and "Record" in the bulleted list above to their respective fields in the form diagram.

Eks. på elementer til bruger-input

Eksempler fra [MDN]

forslagsliste

<datalist>

Choose a flavor:

A screenshot of a dropdown menu. The input field above it says "Choose a flavor:". The dropdown menu contains five items: "Chocolate", "Coconut", "Mint", "Strawberry", and "Vanilla".

options

<input type="radio">

Select a maintenance drone:

- Huey
- Dewey
- Louie

menuer

<select>

A screenshot of a dropdown menu. The input field above it says "Please choose an option--". The dropdown menu contains six items: "Dog", "Cat", "Hamster", "Parrot", "Spider", and "Goldfish".

options

<input type="checkbox">

Choose your monster's features:

- Scales
- Horns

Tekst input/redigering

<textarea>

Tell us your story:

It was a dark and stormy
night...

ranges

<input type="range">

Audio settings:



Validering af inputs

- Validering af input data på klient-side
 - Giver bruger hurtigt feedback (uden server kommunikation)
 - Sparer belastning på server, da den håndterer færre ugyldige anmodninger
 - Sparer båndbredde / data trafik: husk mobildata/roaming kan koste \$€
 - Sparer energi – især vigtigt på mobile / batteridrevne platforme. Kommunikation koster relativt meget energi. CPU typisk meget billigere
- Validering på server-side
 - **Server skal ALTID validere data fra kilder den ikke har 100% tillid fra**
 - Næsten alt eksternt
 - Klienter kan sende hvad som helst
 - være ondsindede
 - fejl-behæftede

HTML5 har faciliteter til basal validering, se https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation
Derudover foretages det vha. JS (kan også give målrettede fejlbeskeder).

Klient side Input Validering

- Kan laves i HTML5 ["forms validation"](#) med dens validerings attributter på input elementer
 - <label for="pauseInput">Indtast pauselænge i sekunder :</label>
<input id="pauseInput" type="number" min="1" max="1200" required name="pause">
- Kan også laves af JS i på klienten
 - Events: "input" og "submit" som fyrer når et felt er indtastet eller form submitted
 - Input felter har normalt en "value" property, som JS kan inspicere
- JS og HTML validering kan kombineres:
 - Fyrer "invalid" event, hvis input ikke matcher krav
pauseInputElement.addEventListener("invalid", invalid);

NewGame Formular

IWP BMI-recorder

Personal information:

Name

Height (cm):

Weight (kg):

```
<form id="bmiForm_id" action="bmi-records" method="post">
<fieldset>
    <legend>Personal information:</legend>

        <label for="name_id"> Name</label>
        <input type="text" name="name" id="name_id" value="Mickey" required minlength="1"
maxlength="30">

        <label for="height_id"> Height (cm):</label>
        <input type="number" name="height" id="height_id" value="" min="1" max="300" required>

        <label for="weight_id"> Weight (kg):</label>
        <input type="number" name="weight" id="weight_id" value="" min="1" max="300" required>

        <input type="submit" id="submitBtn_id" value="Record">
</fieldset>
</form>
```

- Formularens action angiver hvilken URL formularen skal sendes til (nomalt relativt til den aktuelle side: <http://localhost:3000>)
- Formularens action angiver hvilken HTTP metode formularen skal sendes med
- I stedet for placeholder kunne vi have
`value="et default navn"`
- `<input>`'s name attribut og aktuelt input sendes som par (name,value) til server
- `<label>`s for attribut: binder label til `<input>`'s id attribut.
- Validering:
 - Required
 - Input type
 - Min og max (værdier / længder)

Forms: indsendelsesmetode

- ”Method” er typisk GET eller POST
- Se lektion om HTTP

Lidt om CSS

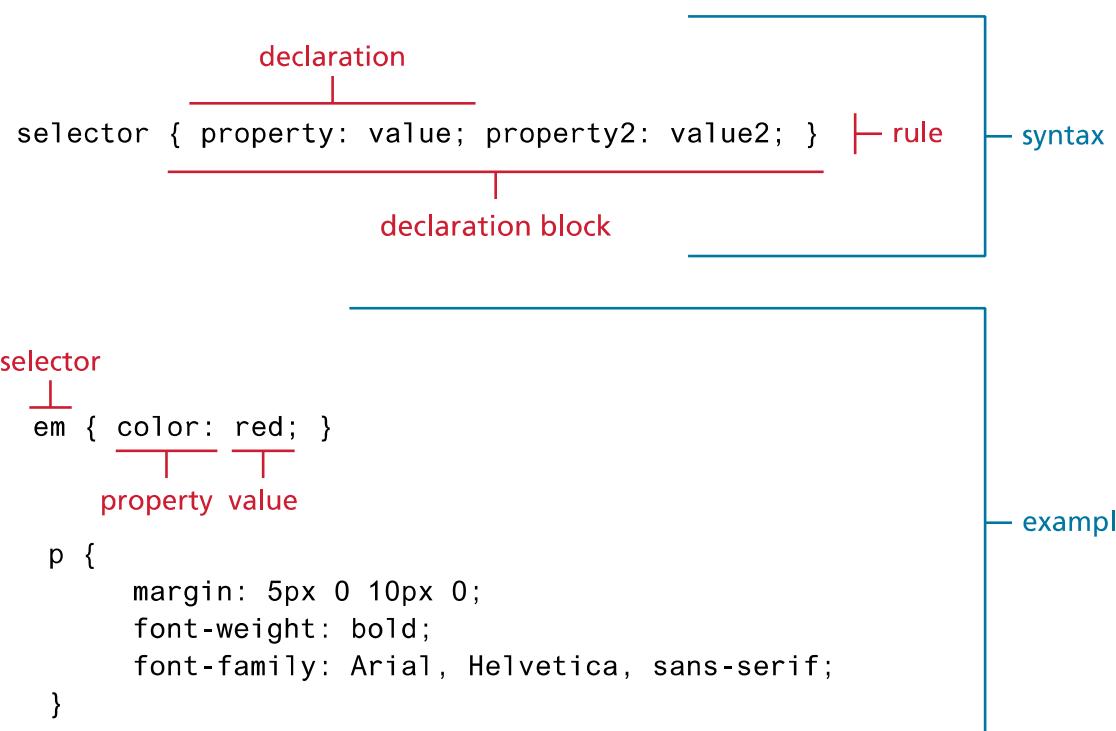
Her er rocker-udgaven! Ellers kan det blive flere virkelig lange forelæsninger

Cascading Style Sheets

- CSS er en W3C standard til at beskrive **udseendet** af HTML elementer
- Tildele egenskaber (properties) som font, farver, størrelser, kanter, baggrunde, og positionering til elementerne på en side
 - Forbedret **kontrol** over formattering.
 - Forbedret **vedligehold** af et site: ret kun ét sted (i css filer).
 - Forbedret **tilgængelighed**: tekst-til-tale for svagtseende forstyrres ikke af layout annotationer
 - Forbedret download **hastighed** (css filer gælder typisk for et helt site, ikke pr side: genbruges fra cache)
 - Forbedret **output fleksibilitet** (responsivt design): kan nemmere tilpasses forskellige klienters skærmstørrelse
- Kan være frustrerende (**historisk uensartet impl i browsere**).
- Layout delen ikke altid intuitiv

CSS regler

- CSS dokument = en liste af stil regler
- En stil regel består af
 - **selector** som udpeger de HTML elementer, som skal påvirkes.
 - En erkæringsblok med en liste af **property:value** par
- Selector kan skrives på flere måder
- Der er mange 100-vise af "properties"



Fx Alle bliver røde og <p> elementer fede med Arial font

CSS properties

Ikke udømmende; CSS er et "rigt" sprog til at udtrykke sådanne egenskaber, og koble dem sammen

| Property Type | Property |
|----------------------|--|
| Fonts | font
font-family
font-size
font-style
font-weight
@font-face |
| Text | letter-spacing
line-height
text-align
text-decoration*
text-indent |
| Color and Background | background
background-color
background-image
background-position
background-repeat
box-shadow
color
opacity |
| Borders | border*
border-color
border-width
border-style
border-top, border-left, ...*
border-image*
border-radius |

| Property Type | Property |
|---------------|--|
| Spacing | padding
padding-bottom, padding-left, ...
margin
margin-bottom, margin-left, ... |
| Sizing | height
max-height
max-width
min-height
min-width
width |
| Layout | bottom, left, right, top
clear
display
float
overflow
position
visibility
z-index |
| Lists | list-style*
list-style-image
list-style-type |
| Effects | animation*
filter
perspective
transform*
transition* |

Selectors (VIGTIGT)

CSS har et rigt sprog til at udpege HTML elementer

- **Type:** udvælger elementer med et givet element navn
- **Class:** udvælger alle elementer med det givne klasse-attribut ":"
- **ID:** udvælger ud fra id-attribut "#"
- **Attribut:** udvælger [attribut] {...}
- **Pseudo:** udpeger på baggrund af et elements tilstand
`a:link, a:visited, :focus, :hover, :active ...`
- **Kontekstuel:** baseret på naborelation i HTML-træet.

| Selector | Example | Learn CSS tutorial |
|-----------------------------|--------------------------------|------------------------|
| Type selector | <code>h1 { }</code> | Type selectors |
| Universal selector | <code>* { }</code> | The universal selector |
| Class selector | <code>.box { }</code> | Class selectors |
| id selector | <code>#unique { }</code> | ID selectors |
| Attribute selector | <code>a[title] { }</code> | Attribute selectors |
| Pseudo-class selectors | <code>p:first-child { }</code> | Pseudo-classes |
| Pseudo-element selectors | <code>p::first-line { }</code> | Pseudo-elements |
| Descendant combinator | <code>article p</code> | Descendant combinator |
| Child combinator | <code>article > p</code> | Child combinator |
| Adjacent sibling combinator | <code>h1 + p</code> | Adjacent sibling |
| General sibling combinator | <code>h1 ~ p</code> | General sibling |

[https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors#in this module](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors#in_this_module)

CSS Ex. Tabel farvning

IWP Multi Yatzy

Play Game:

Yatzy Scores

| Peter | | |
|-----------------|------|-------|
| Round Name | Dice | Score |
| 1s | | 1 |
| 2s | | 0 |
| 3s | | 0 |
| 4s | | 0 |
| 5s | | 0 |
| 6s | | 0 |
| Sum | | 1 |
| Bonus | | 0 |
| 1 Pair | | 0 |
| 2 Pairs | | 0 |
| Three Identical | | 0 |
| Four Identical | | 0 |
| Little Straight | | 0 |
| Big Straight | | 0 |
| House | | 0 |
| Chance | | 0 |
| Yatzy | | 0 |
| Total Score | | 1 |

<h1>

id="scoretable"

Header celler indenfor
id="scoretable"

class="rowfill"

Mus svæver over rækken

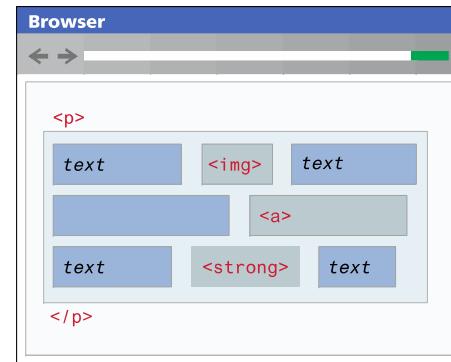
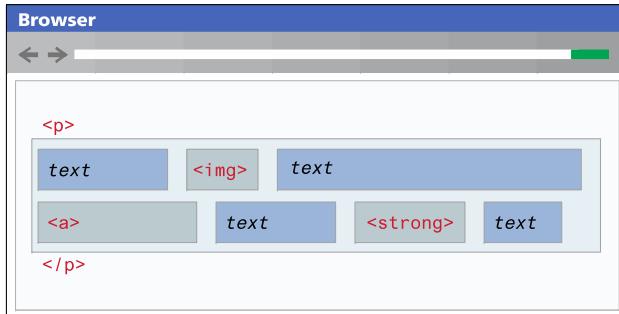
```
h1 {  
    background-color: lightgreen;  
}  
  
#scoretable {  
    font-family: Arial, Helvetica, sans-serif;  
    border-collapse: collapse;  
    border: 2px solid green;;  
    color: black;  
}  
  
#scoretable th {  
    padding-top: 12px;  
    padding-bottom: 12px;  
    text-align: center;  
    color: green;  
    border: 2px solid green;  
}  
  
#scoretable .row-fill {  
    background-color:#a1dfa3;  
    border: 2px solid green;;  
    padding: 8px;  
}  
#scoretable tr:hover {background-color: #ddd;}  
...
```

Kaskade-regler: Hvordan styles interagerer

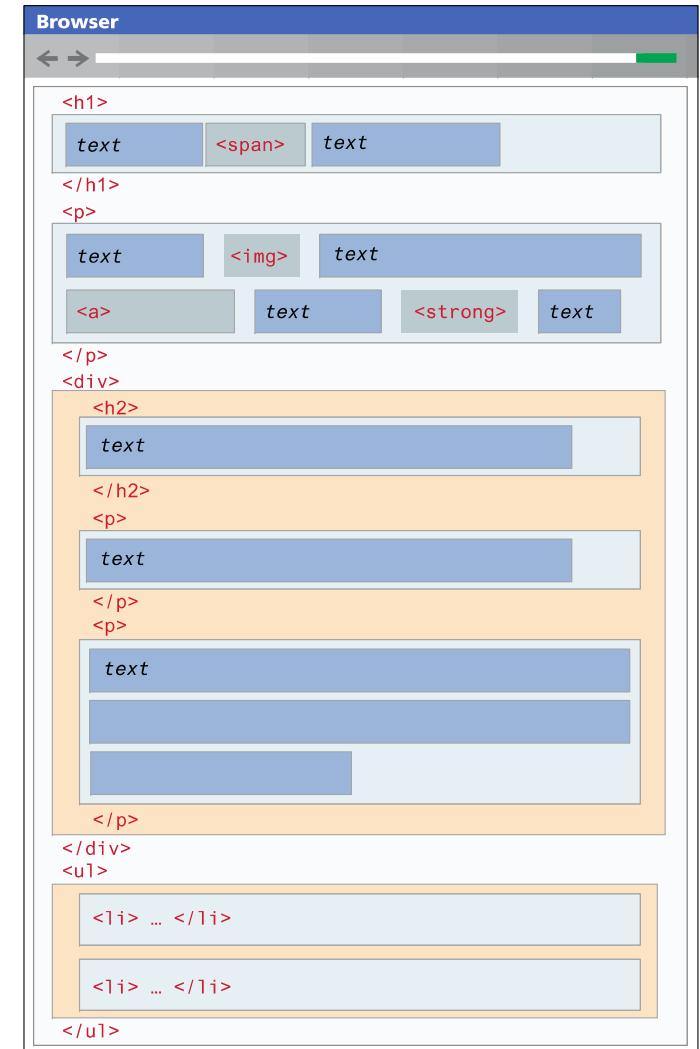
- “Cascade” i CSS henviser til algoritmen for hvordan stil-reglerne sættes sammen til en rendering af et element.
 - Definerer også hvordan overlappende/konfliktende regler håndteres. 1 er højst prioritet
 - 1. Oprindelse: Der kan være flere stylesheets i spil
 1. Slut-bruger kan have eget style-sheet (fx syns-handicappet)
 2. Web-forfatterens style-sheet
 3. Browser's indbyggede default;
 - 2. Specificitet: Des mere specifik selektor, des højere prioritet
 - 3. Regler med samme specificitet: erklæringsrækkefølge af regler er betydende så sidst erklærede vinder
 - 4. Nedarvning : ”arvelige” CSS egenskaber påført et HTML element videreføres automatisk også på elementets børn.
 - Arvelige CSS egenskaber: fx font, color, text
 - Ikke-arvelige egenskaber: fx layout, størrelser, kanter, baggrunde,
- https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance
- <https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade>
- <https://blog.logrocket.com/how-css-works-understanding-the-cascade-d181cd89a4d8/>

CSS positionering og layout: Normal flow

- HTML dokumentet læses fra top mod bund
- **Block-niveau elementer** (`<p>`, `<div>`, `<h2>`, ``, og `<table>`) får deres egen linie: top mod bund
 - Nogle kan nestes
 - I forældre elementets ramme
- **Inline elementer** (``, `` vises på samme linie venstre mod højre
 - Resize=>rewrap

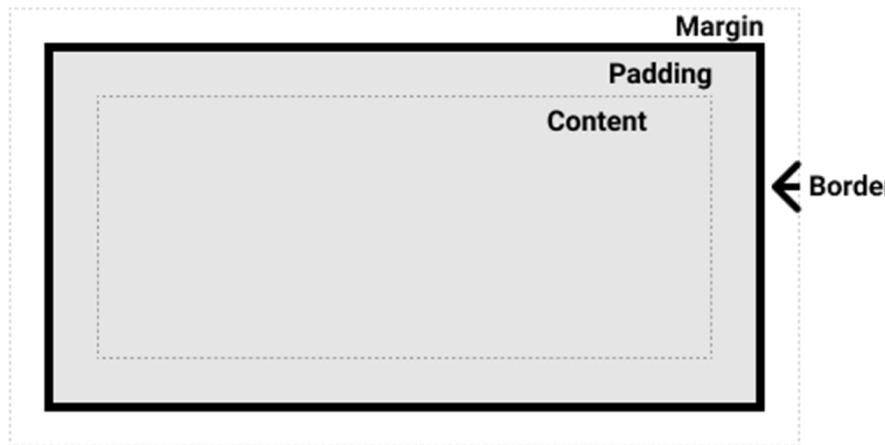


- `` og `<div>` anvendes hyppigt! Tillader meget fin-kornet kontrol af layoutet
- SEMANTISK MARKUP BØR ANVENDES HVOR MULIGT

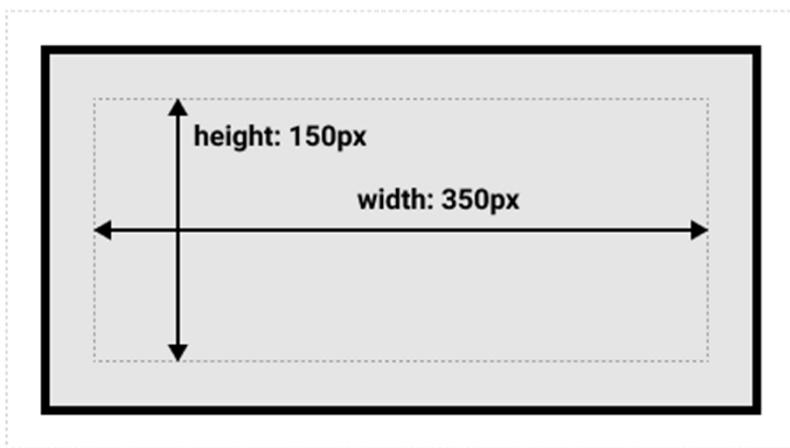


CSS positionering og layout: box-modellen

- Alle HTML elementer renderes i en "box"



```
1 .box {  
2   width: 350px;  
3   height: 150px;  
4   margin: 10px;  
5   padding: 25px;  
6   border: 5px solid black;  
7 }
```



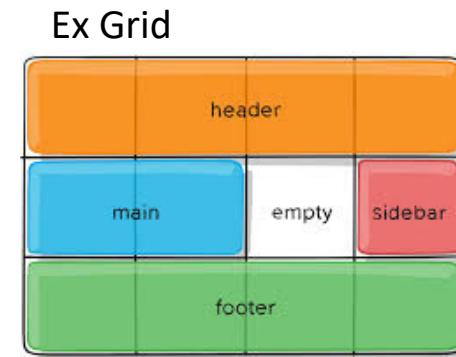
"Inspector" i "developer tools"

A screenshot of the Chrome DevTools Inspector. The 'Styles' tab is selected, showing CSS rules for the body element. On the right, the 'Elements' panel displays a hierarchical tree of the page's DOM structure, with each node representing a box and its dimensions. The tree shows the nested layers of the box model: margin, border, padding, and content. The content box is highlighted in blue, and its dimensions (1164x30534) are visible. The 'Properties' tab is also visible at the top.

```
html body.logged-out.env-production.min-width-lg  
body {  
  word-wrap: break-word;  
}  
body {  
  font-family: -apple-system,BlinkMacSystemFont,Segoe UI,Helvetica,Arial,sans-serif,Apple Color Emoji,Segoe UI Emoji;  
  font-size: 14px;  
  line-height: 1.5;  
  color: #24292e;  
  background-color: #fff;  
}  
body {  
  margin: 0;  
}
```

CSS positionering og layout: styring af bokse

- CSS indeholder en række mekanismer til layout af bokse
- Display egenskaben:
 - Vælger layout for elementets børn, fx
 - display: flow Normalt flow layout
 - display: grid Opstilling på basis af en matrice/"skelet"
 - display: flex Automatisk opstilling vertikalt eller horisontalt: alignment, justering af mellemrum, wrapping følger
 - Se også https://www.w3schools.com/css/css3_flexbox.asp
 - display:block, display:inline Vælger om elementet skal behandles som inline eller block
 - Om elementet skal vises eller ej (styres typisk via Javascript)
 - #div1{display: none}
 - #div1{visibility:hidden} (optager plads)
- Position egenskaben: (x,y) koordinater
 - Fin-justering af et elements position ifht. forventet placering
 - Adskillige modes(fixed, relativt, absolut,sticky..)



CSS Ex. Layout og farvelader

Not secure | http://people.cs.aau.dk/~bnielsen/links.html

IWP Demo Page

Contents Outline:

- [Introduktion](#)
 - [HTML](#)
 - [JavaScript](#)
- to do

Introduktion til Web-Programmering

Centrale elementer i web-teknologi er [HTML](#) og [JavaScript](#).
la bla bla.
Klient-side programmering kommer i [næste lektion](#).
Her er en god [reference til web-teknologi](#).

HTML

Med HTML kan du lave semantisk mark-up af et dokument, som vises i en web-browser. Etc. Etc. Etc.

HTML BODY

Indeholder vigtig information om dokumentet, som modtageren skal bruge for at forstå det.

JavaScript

JS kan bringes til at fungere sammen med en browser og interagere med en HTML side.



Fig. 1 - Sød Panda som browser bambus! © National Geographic

According to Goodreads, [Turing Award Winner Edsger Dijkstra](#), has stated that " If debugging is the process of removing software bugs, then programming must be the process of putting them in. [https://www.goodreads.com/author/quotes/1013817.Edgeser_W_Dijkstra]
Du skal dog først kende til [HTML](#).

No remarks

```
body {  
    display: grid;  
    grid-template-areas:  
        "h h h"  
        "n m a"  
        "f f f";  
    grid-template-rows: 70px 1fr 60px;  
    grid-template-columns: 20% 1fr 15%;  
    grid-row-gap: 10px;  
    grid-column-gap: 10px;  
    height: 100vh;  
    margin: 0;  
}  
/*general coloring and layout */  
header, footer, main, nav, aside {  
    padding: 0.5em;  
    background: lightgray;  
    margin: 0.5em;  
    color: black; box-shadow: 0 0 0.25em black;  
}  
header { grid-area: h; }  
footer { grid-area: f; }  
main { grid-area: m; }  
nav { grid-area: n; }  
aside { grid-area: a; }  
header{  
    background-color: lightgreen;  
}  
  
section{  
    margin: 1em;  
    border: thin dashed; padding: 1em;  
    background: rgb(190, 190, 190);  
}  
#introsection{ background: rgb(248, 229, 55);}  
main{  
    /* try to uncomment this */  
    /*display: flex;*/  
}
```

Forfattet af Brian Nielsen

Angivelse af stil-regler

1. Som ekstern .css text fil
 - den normale, anbefalede metode.
2. Indlejret metode: i HTML som style element i header
 - Ved kort style-sheet
3. Direkte metode via “style” attribut på HTML elementet).
 1. Anvendelse bør minimeres!
 2. Kan være handy til test af ny stil
 3. Gælder kun på det konkrete element

```
<head> ...
  <link rel="stylesheet" href="styles.css">
</head>
```

```
<head> ...
  <style>
    h1 { font-size: 24pt; }
    h2 {
      font-size: 18pt;
      font-weight: bold;
    }
  </style>
</head>#
<body>
```

```
<h2 style="font-size: 24pt; font-weight:bold;">
Karakterer</h2>
```

CSS Layout Strategier

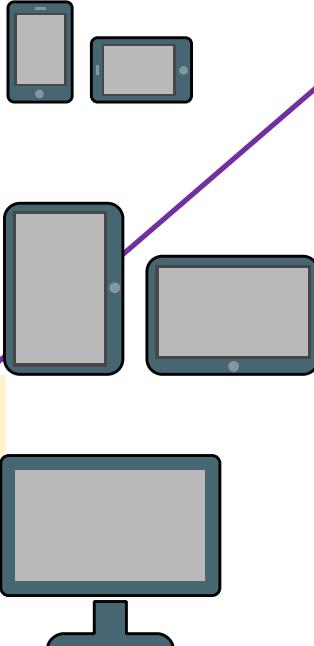
- **Fast layout:**
 - Web-designer vælger én ideal/typisk skærmstørrelser og tilpasser designet til den.
 - Simplere og mere forudsigtigt resultat (for den ene valgte størrelse)
- **Liquid-layout:**
 - Alle mål angives relativt i %, og browser renderer til den givne klient
 - Tricky at få pænt til alle tilfælde
- **Responsivt-web-design**
 - Layout tilpasser sig forskellige klienters (desktop, tablet, mobil, printere) vindue- og skærm-størrelser, og stående eller liggende (portrait/landscape)
 - Mere avanceret, mere arbejde
 - CSS har mekanismer til formålet
 1. Indstilling af klientens synsfelt (“viewports”) via `<meta>` element
 2. Lav versioner af CSS afhængigt af klientens “viewport” vha. forspørgsler på skærmstørrelse (“media queries”)
 3. Brug principper for liquid layouts, og skalér af billeder til størrelsen af “viewport”

Responsivt Web-design

- **Viewport:** området i en web-sider hvor indhold er synligt for brugeren

- For stor web-side: scroll-bars
- For lille: skærm udnyttes ikke
- Typisk optimeret indstilling:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0">
```



styles.css

```
/* rules for phones */  
@media only screen and (max-width:480px)  
{  
    #slider-image { max-width: 100%; }  
    #flash-ad { display: none; }  
    ...  
}  
  
/* CSS rules for tablets */  
@media only screen and (min-width: 481px)  
and (max-width: 768px)  
{  
    ...  
}  
  
/* CSS rules for desktops */  
@media only screen and (min-width: 769px)  
{  
    ...  
}
```

- **Medie forespørgsler**

Instead of having all the rules in a single file, we can put them in separate files and add media queries to `<link>` elements.

```
<link rel="stylesheet" href="mobile.css" media="screen and (max-width:480px)" />  
<link rel="stylesheet" href="tablet.css" media="screen and (min-width:481px)  
and (max-width:768px)" />  
<link rel="stylesheet" href="desktop.css" media="screen and (min-width:769px)" />
```

CSS Resourcer, biblioteker, og værktøjer

- <https://css-tricks.com/>

- Der findes en række biblioteker (frameworks) med CSS des skabeloner

- <https://learnlayout.com/frameworks.html>

- Der findes design værktøjer og CSS "generatorer"

- <https://www.creativebloq.com/features/best-web-design-tools>



Råd om CSS til projektet

- KISS
 - Start med absolut minimal styling og layout, god funktionalitet
 - Start med fast layout til en "normal" desktop skærm
 - Ambitions-niveauet kan stige, men det kan være en tidsrøver
 - Start med vanilla JS/HTML/CSS, når i mestrer principperne i dette kan i overveje komponenter a la bootstrap.
 - Der er ikke specifikke læringsmål der går på fancy grafik og layout!
(men det må selvfølgeligt gerne se rimelig ud)

Grundlæggende HTTP

HyperText Transfer Protocol

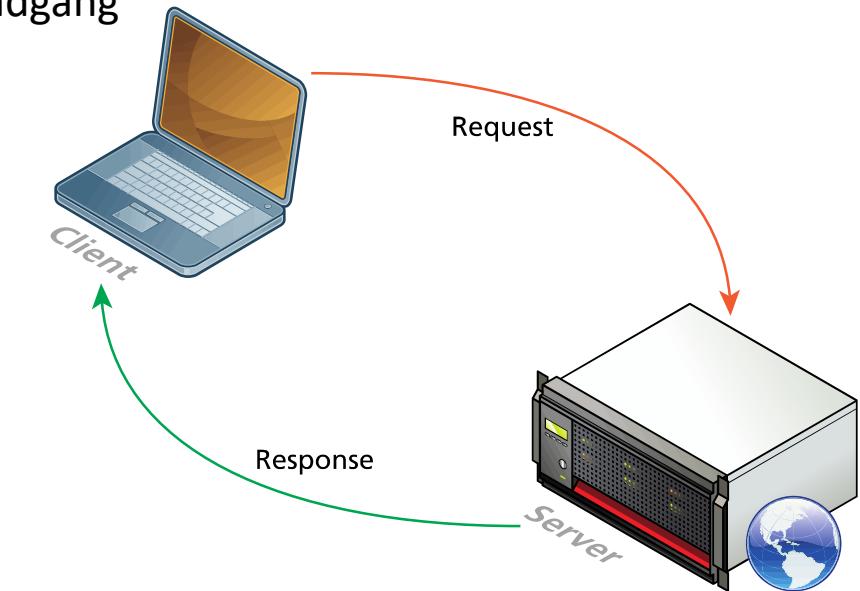
Client-Server Modellen

- Generelt:

- Server tilbyder en service, som klienten gerne til have adgang til
 - Indkapsler en ressource (data, hardware, funktionalitet)
- Klient er initiativ tager til kommunikation
 - Typisk Request-respons mønster
- Server er typisk passiv til adspurgt
- (server kan være klient overfor andre servere)
- En server har mange, flygtige, klienter
- En server er "always on"
- Kendt DNS Navn

- I WWW:

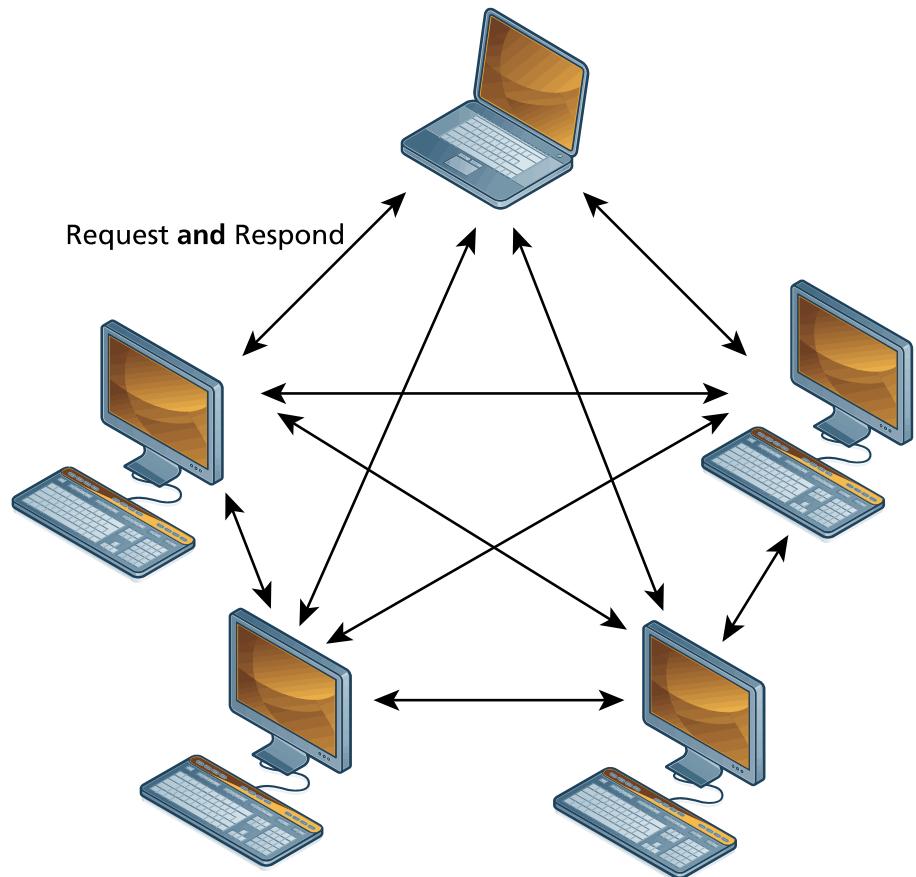
- Klienten er oftest en web-browser applikation
- "Server" kan være web-server-maskiner i store installationer
- Anvender typisk HTTP som request-respons protokol
- Data leveret i form af HTML (eller andet web-format som JSON, XML)
- Server gemmer applikationens tilstand (persistent)



Fil server
Print server,
Web server,
...

Sidebemærkning: Peer-to-peer modellen

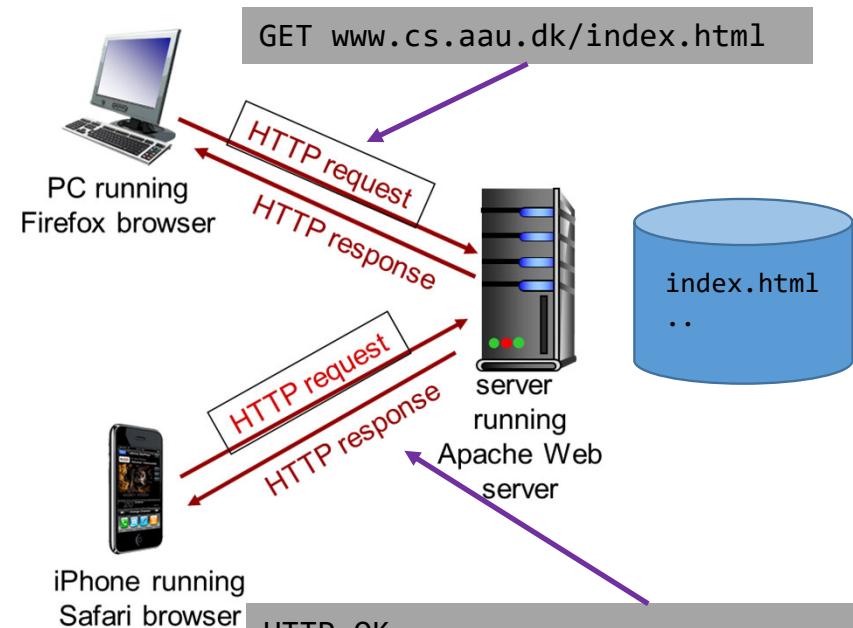
- Modsætning til client-server
- Symmetrisk ansvar



Simpel HTTP Eksempel Scenarie: resource er statisk html fil

HTTP: hypertext transfer protocol

- applikationslags-protokol til web-trafik
- client/server model:
- **client:** program (typisk browser)
 1. Sender forespørgsel (vha. HTTP GET), om en web side
 2. Afventer respons
 3. Parser respons og optegner siden
 4. Kører evt. skridt 1-3 igen (sideløbende) for at henter nødvendige indlejrede ressourcer (billeder, js-scripts, style sheets)
- **server:**
 1. Afventer
 2. Modtager HTTP forespørgsel fra klient
 3. Behandler den, og læser filen,
 4. Sender HTTP respons med fil-indhold til klienten
- Server og ressourcen angives ved et Uniform Ressource Identifier, normalt URL



```
HTTP OK
...
<html>
  <head>
    <link> ... "my.css" </link>
    <script>... "my.js" </script>
  </head>
  <body>
    <h1> Computer Sc. </h1>
    
  </body>
<html>
```

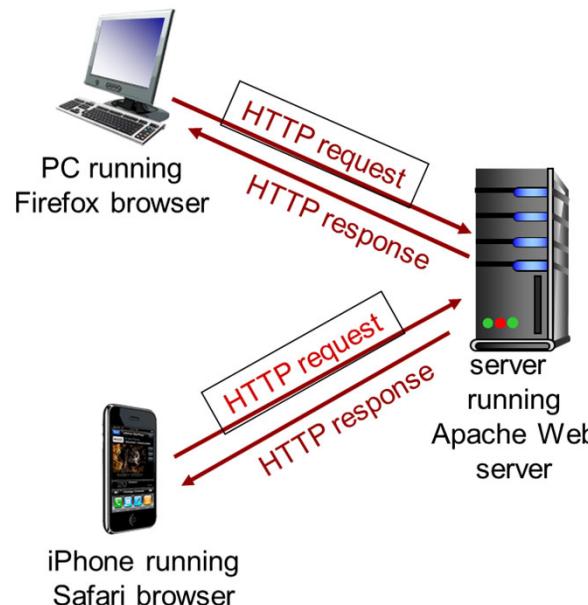
Det eksakte forløb afhænger af HTTP protokol version
(jfv. Forelæsning om applikationslagsprotokoller)

HTTP Generel Oversigt

HTTP: hypertext transfer protocol

- applikationslags-protokol til web-traffik
- client/server model
 - **client:** program (typisk browser) som anmoder (vha. HTTP), modtager svaret, og behandler svaret
 - **server:** Web server modtager anmodningen, beregner svaret, og sender (vha HTTP) resultatet til klienten
- Server er identificeret vha.
 - en IP-adresse (eller DNS navn) og et port nummer på serveren,
 - port nummer: 80 eller (443 for HTTPS)

Alternativt klient program: "CURL"



Server er værtsmaskine for et antal **ressourcer**:
en eller anden "ting" på serveren

- Information, data i db
- Web sider
- Filer
- Billeder
- Beregninger
- Services
- IoT Device / fx temperatur måling

En ressource har en eller flere **repræsentationer (dokument)** som afspejler dens aktuelle indhold/tilstand på en måde som kan sendes over netværk

- Metadata
- Sekvens af bytes
- Når server modtager en HTTP forespørgsel, kører den kode, som beregner svaret ("repræsentationen")
 - Dynamiske web sider, DB opslag
 - Program som server ejerne har lavet (fx jeres program)
 - REST APIs

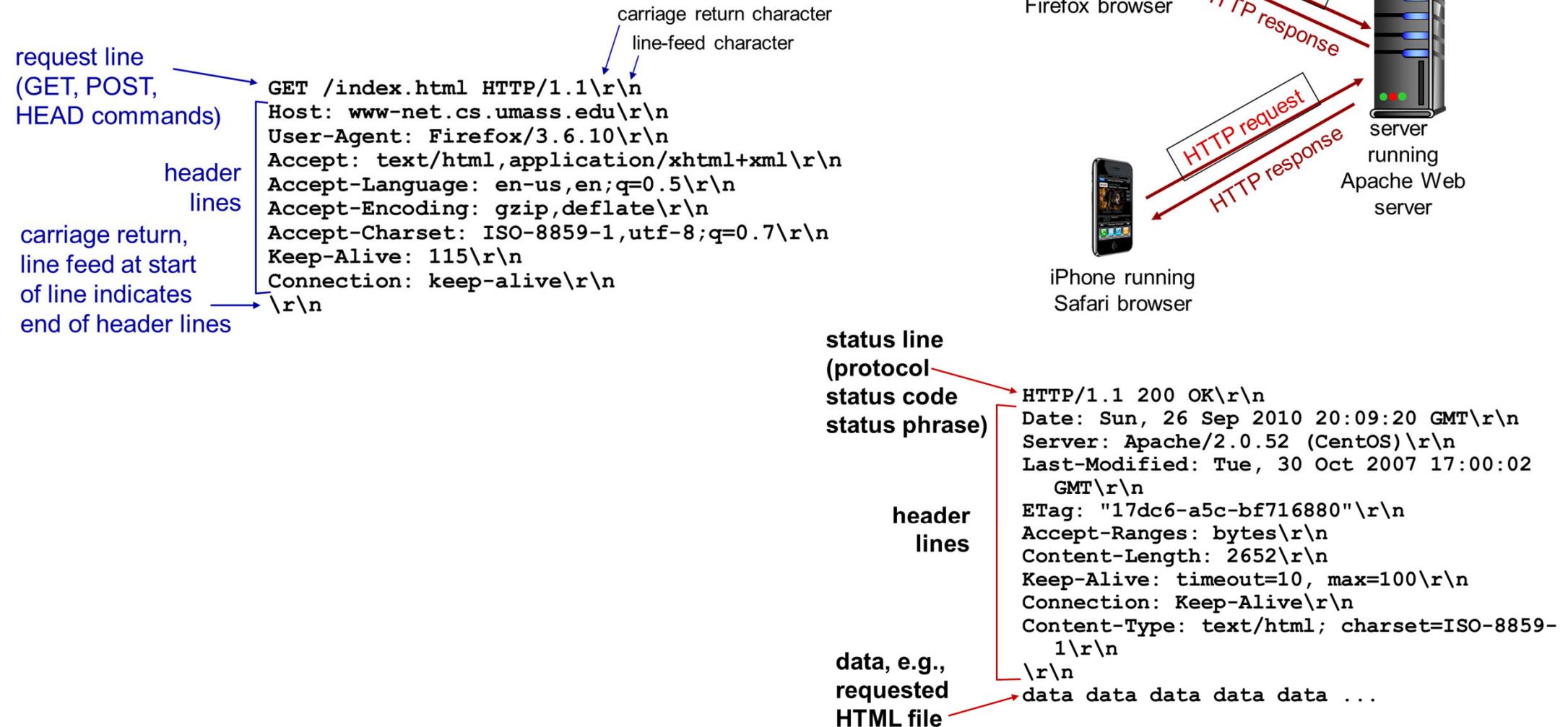
HTTP Besked

- En besked (både request og respons) består af 2 dele:



- Header: formål med beskeden, og kontrol information om beskeden, protokol flag, optioner
- Body: de egentlige data, der udveksles
 - Data kan være hvad som helst (html, json, billeder, fil-data,)

HTTP Beskeder



HTTP/1.1 Metoder (a.k.a "verbs")

- **GET:**

- Anmoder om overførsel af en repræsentation af den ønskede ressource
- "Læsning"

- **POST:**

- Udfør en resource-specifik behandling på den ønskede ressource
- "Ændring"
 - Fx, tilføje data til ressourcen (fx indtastet i en "HTML form")

- **PUT:**

- Oprette (eller erstatte) tilstanden på den ønskede ressource) i sit hele så den svarer til den medsendte repræsentation

- **DELETE**

- Sletter ressourcen (eller fjerner forbindelsen imellem URL navn og ressourcen)
- HEAD, PATCH, CONNECT, OPTIONS, TRACE

Når serveren modtager et request med en given metode kalder den en **funktion** som "du" eller (web-server programmøren) har lavet!

"Spilleregler" for web-arkitekturen forudsætter at **funktionen** (som udføres på server siden) skal respektere hensigten bag HTTP metoderne

- Fx må GET ikke ændre ressourcen.
- Caching, forudindlæsning ("pre-fetching")

Sikre metoder: udførslen af **funktionen** må ikke "skade" ressourcen, eller give unormal stor belastning på server

- Klient kan gentage dem
- GET, HEAD, OPTIONS, TRACE skal programmeres så de er sikre
- Et PUT umiddelbart efterfulgt af et GET skal give den værdi der netop er oprettet
- Begrebet "**Idempotente**" operationer introduceres senere

DIN KODE SKAL RESPEKTERE DISSE

Content-Type header

- Indholdet af HTTP Body bestemmes af “Content-Type” feltet i HTTP headeren
 - Modtageren (fx browser) bruger den til at afgøre hvordan dataene skal behandles
 - Content-type skal der angives korrekt: MIME type Multi-purpose Internet Mail Extensions
 - Fil-endelser kan bruges af sender til at “gætte” en MIME-type

Generelt format:
type/subtype;parameter=value

Fil Endelse	Dokument art	MIME Type	Eksempel
.bin	Any kind of binary data	application/octet-stream	text/plain; charset=UTF-8
.css	Cascading Style Sheets (CSS)	text/css	
.htm .html	HyperText Markup Language (HTML)	text/html	
	Html formular data	application/x-www-form-urlencoded	
.jpeg .jpg	JPEG images	image/jpeg	
.js	JavaScript	text/javascript	
.json	JSON format	application/json	
.mjs	JavaScript module	text/javascript	
.mp3	MP3 audio	audio/mpeg	
.mpeg	MPEG Video	video/mpeg	
.png	Portable Network Graphics	image/png	
.pdf	Adobe Portable Document Format (PDF)	application/pdf	

HTTP Responskoder

1. 100-199: Informativt svar
2. 200-299: Succesfuld respons
3. 300-399: Omdirigering (Redirects)
4. 400-499: Client fejl
5. 500-599: Server fejl

102 Processing

- Server arbejder, tålmodighed, tak

200 OK

- Anmodning gik godt!
 - GET: resourcen er hentet og følger i beskedens “body”
 - POST: handling gik godt og resultatet følger i beskedens “body”

301 Moved Permanently

- Det forespurgte objekt er blevet flyttet; ny placering følger senere i beseden

400 Bad Request

- Forespørgslen kunne ikke forståes af server

404 Not Found

- Den forespurgte resource kunne ikke findes på denne server

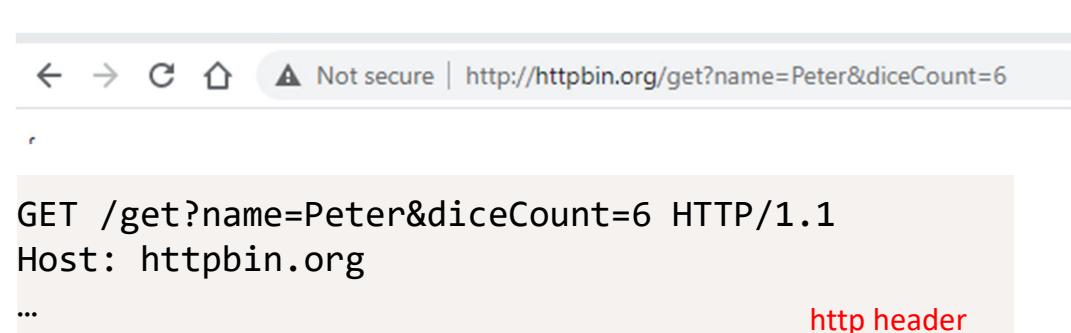
505 HTTP Version Not Supported

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

HTTP og FORMS: GET

- **GET:** Kan bruges ved læsning af ressourcen
- Søge parametre kan sendes som en del af URL'en (i "query string")
 - kan ses direkte af bruger i browser
 - Indgår i browser historik, kan bogmærkes
 - Respons kan cache's
 - Query string kan kun være 2048 karakter lang
 - URL-encodes
 - Kan ikke indeholde binær data
- Form data sendes som i "query string"
 - Bemærk "name" attribut til identifikation af parametrenes navne

```
<form action="http://httpbin.org/get" method="get">
  <input type="text" id="name_id" name="name" >
  <input type="number" id="diceCount_id" name="diceCount" >
  <input type="submit" value="New Game">
</form>
```



(httpbin.org er et web-site til test af web-requests)

HTTP og FORMS: POST

- **POST:** Anvendes normalt til ændring af ressourcen (se mere specifikt slides http "verbs")
- Form data sendes som del af request-body
 - Url-encoded, jfv. Content-Type
 - Indgår ikke i historik, cache
 - Gensendes ikke af browser uden advarsel (js kan dette)
 - Kan (kun) under specielle omstændigheder anvendes til læsning
- (Fil-upload: <input type="file"> skal overføres som content type "multipart/form-data")

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type>

```
<form action="http://httpbin.org/post" method="POST">
  <input type="text" id="name_id" name="name" >
  <input type="number" id="diceCount_id" name="diceCount" >
  <input type="submit" value="New Game">
</form>
```

POST /post HTTP/1.1

http header

Host: http://httpbin.org

Content-Type: application/x-www-form-urlencoded

Content-Length: 22

name=Peter&diceCount=7

http body

```
<form action="/" method="post" enctype="multipart/form-data">
  <input type="text" name="description" value="some text">
  <input type="file" name="myFile">
  <button type="submit">Submit</button>
</form>
```

END