

Javascript og DOM

Martin Bregnhøj
mabe@kea.dk

Dagens mål

1. At forstå dokument strukturen (DOM)
2. og være i stand til at ændre på indholdet

Agenda

- DOM (Document Object Model)
- HTML struktur recap
- Vælg et DOM-element i javascript
- Læg nyt indhold i DOM-element
- EventListeners og eventhandlers på DOM-element

DOM

*The Document Object Model (DOM) is an application programming interface (API) for valid HTML ... documents. It defines the logical structure of documents and the way a document is **accessed** and **manipulated**... ...With the Document Object Model, programmers can documents, navigate their structure, and add, modify, or delete elements and content. **Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model***

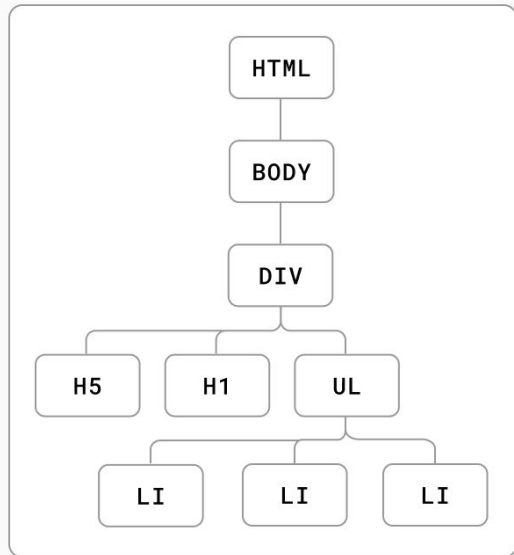
w3.org

DOM træ

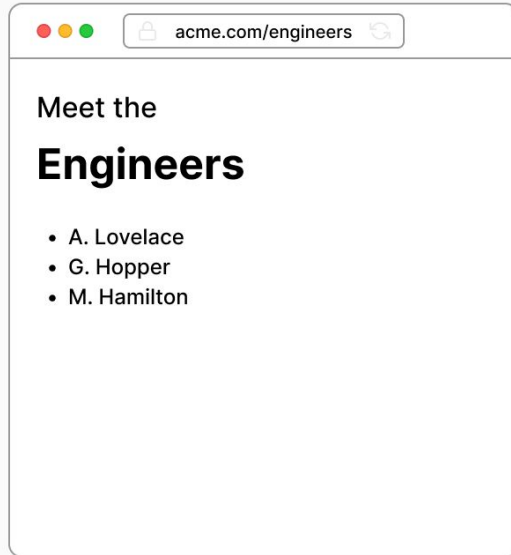
HTML

```
index.html
1 <html>
2   <body>
3     <div>
4       <h5>Meet the</h5>
5       <h1>Engineers</h1>
6       <ul>
7         <li>A. Lovelace</li>
8         <li>G. Hopper</li>
9         <li>M. Hamilton</li>
10      </ul>
11    </div>
12  </body>
13 </html>
14
15
16
```

DOM



UI



HTML struktur recap

HTML5 - Semantiske tags

header, nav, main, section, article, footer, details, summary etc.

Hvad er semantiske tags?

Hvad gør de godt for? - bl.a. SEO

Hvordan skal de bruges?

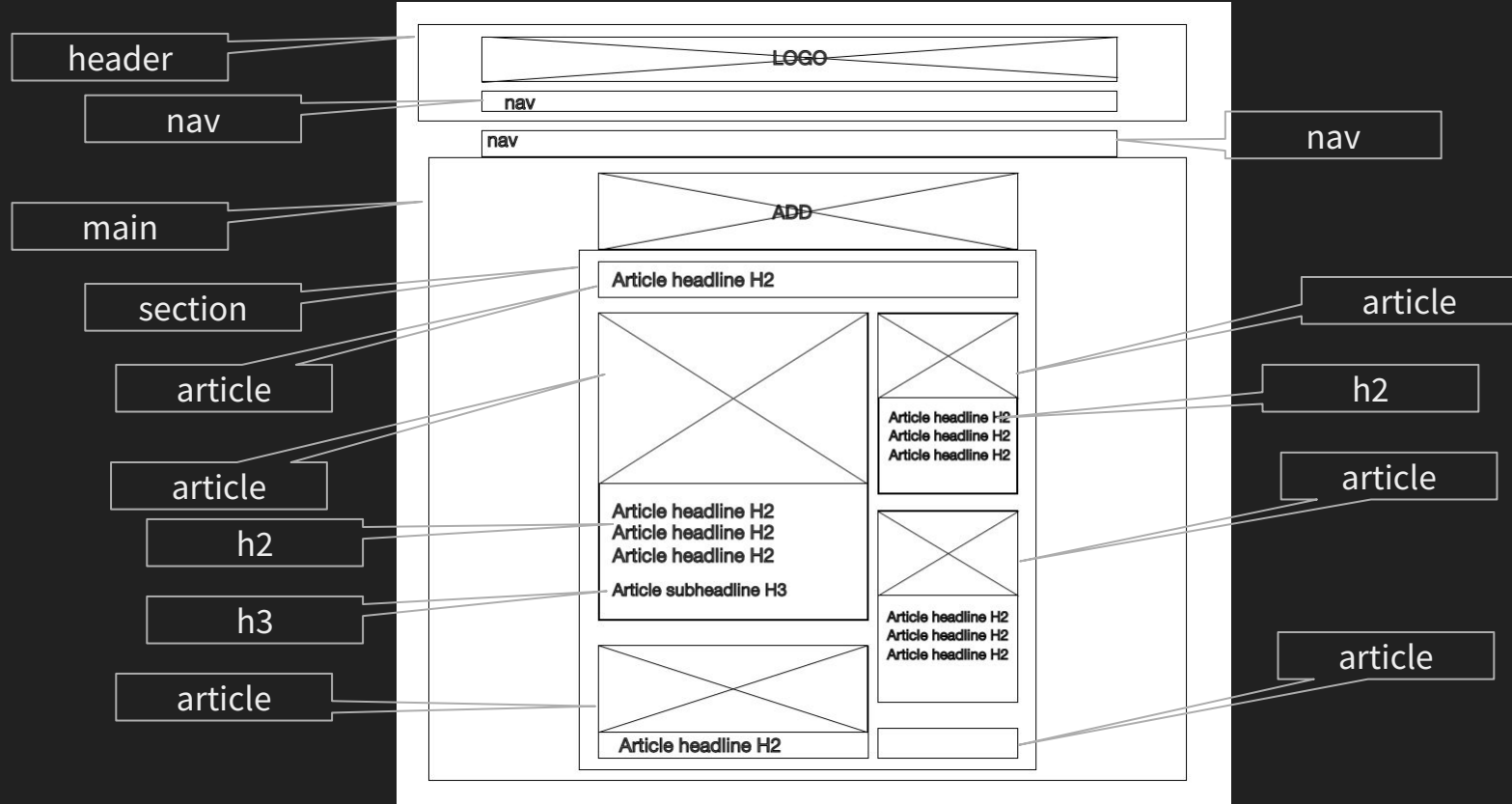
https://www.w3schools.com/html/html5_semantic_elements.asp

[https://developer.mozilla.org/en-US/docs/Glossary/Semantics#Semantics in HTML](https://developer.mozilla.org/en-US/docs/Glossary/Semantics#Semantics_in_HTML)

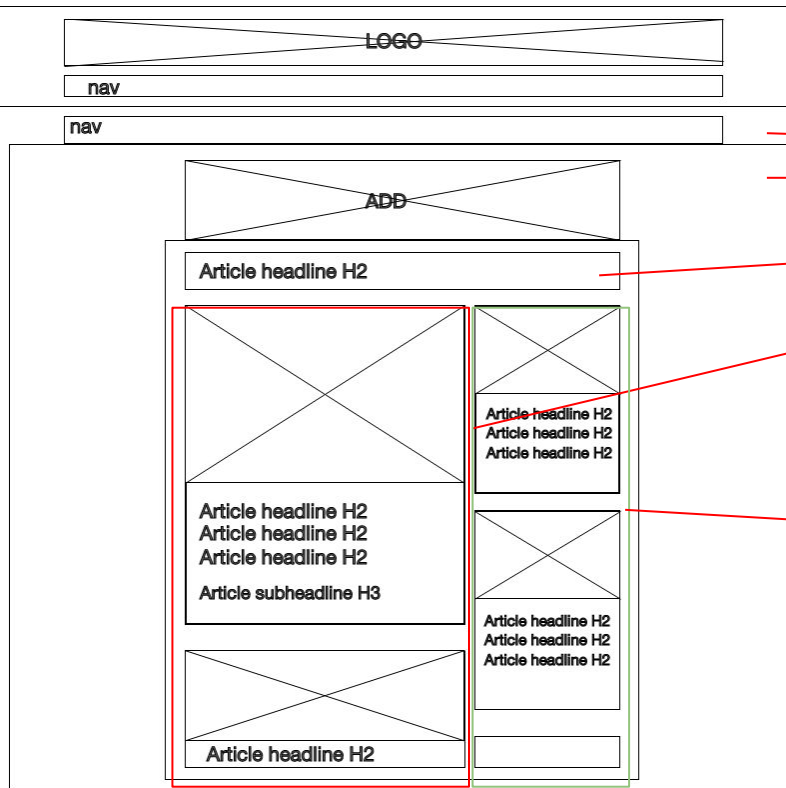
Semantisk html struktur



Wireframe (lettere forenklet)



WIREFRAME



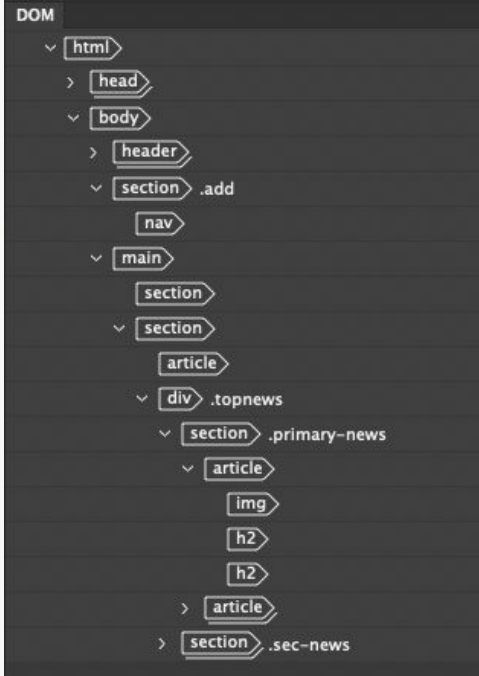
HTML

```

10 <body>
11   <header>
12     <img src="" alt="">
13     <nav></nav>
14   </header>
15   <section>
16     <nav></nav>
17   </section>
18   <main>
19     <section></section>
20     <section>
21       <article class="toparticle"></article>
22       <div class="topnews">
23         <section class="primary-news">
24           <article>
25             <img src="" alt="">
26             <h2></h2>
27             <h3></h3>
28           </article>
29           <article>
30             <img src="" alt="">
31             <h2></h2>
32             <h3></h3>
33           </article>
34         </section>
35         <section class="sec-news">
36           <article>
37             <img src="" alt="">
38             <h2></h2>
39           </article>
40           <article>
41             <img src="" alt="">
42             <h2></h2>
43           </article>
44         </section>
45       </div>
46     </section>
47   </main>
48 </body>

```

DOM TREE



Begreber

- DOM (Document Object Model) - strukturen i et HTML dokument
- Semantisk HTML - html elementer der indikerer hvad de indeholder (godt for SEO og struktur)

In the HTML DOM (Document Object Model), everything is a node:

- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

Tips til øvelse 1

Billeder kan autogenereres på:

<https://picsum.photos/>

Definer en størrelse og indsæt url'en som src. Attribut.

Ex.: ``

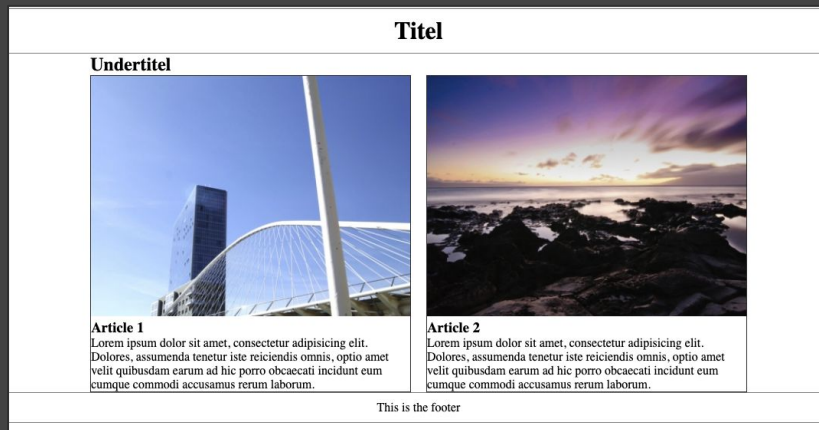
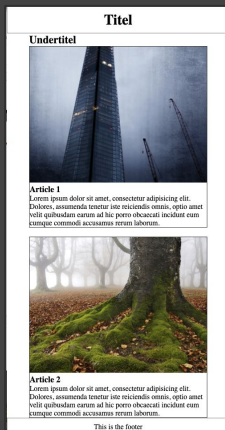
Se dokumentationen på sitet for flere muligheder!

Vælg f.eks. Et bestemt billede ved at tilføje billedets id:

``

Øvelse 1 - HTML5 struktur

1. lav et nyt (git) projekt til øvelserne i VS Code.
2. Opret en ny html-fil.
(tip! gem et tomt doc som html, og brug EMMET: **!** - **tab** til at lave et nyt HTML skelet)
3. Opbyg nedenstående layout med semantiske tags. Billederne vælger du selv.
4. Commit og push til GitHub



Note!

Eks. på overordnet struktur:

- header
- main
 - section
 - article
 - article
- footer

Brug flex eller grid (eller begge dele) til at placere elementerne og indholdet.

Det er ikke nødvendigt at bruge media queries!!!

Vælg DOM-element


document.querySelector - (vælg et DOM-element)

Vælg et element ud fra:

ID: `document.querySelector("#idNavn")`

Class: `document.querySelector(".classNavn")`

Element: `document.querySelector("nav .menupunkt a")`



Alle CSS-selectors kan bruges!

Eksempler på pseudo-selectors:

`document.querySelector("article:first-child")`

`document.querySelector("article:nth-child(2)")`

Er der flere elementer med samme class eller tag, vælges kun det første element!

- Hvis man vil have dem alle? - det vender vi tilbage til!

HTML DOM `querySelector()` Method, w3Schools: https://www.w3schools.com/jsref/met_document_queryselector.asp

Gem querySelector i variabel

Et **DOM-element** valgt med querySelector gemmes tit i en **konstant**, for at man ikke skal gentage selektionen:

```
const info = document.querySelector("#mitElement");  
console.log(info);
```

Hver gang dette element skal bruges i programmet, kan man nu bruge **info**.

Læg nyt indhold i
DOM-element

Nyt indhold i element (textContent og innerHTML)

Empty-tags ex.: img, br, hr, input

containertags ex.: body, article, div, h1, p (har slut-tags)

Eksempel på containertag med indhold: `<footer id="info" >her stå noget</footer>`

Man kan ændre på indholdet af containertags med js:

```
1 const info = document.querySelector("#info");
2 info.textContent = "noget helt andet";
3 info.innerHTML = "<h1>Noget nyt og anderledes</h1>"
```

W3schools: HTML DOM textContent Property, : https://www.w3schools.com/jsref/prop_node_textcontent.asp

W3schools: HTML DOM innerHTML Property, : https://www.w3schools.com/jsref/prop_html_innerhtml.asp

Nyt indhold i element (attributter)

Tags der er afhængige af attributter ex.: img(src), a(href)

Eksempel på attributter: ``

Man kan ændre på indholdet af attributter med js:

```
const pic = document.querySelector("img");  
pic.src = "etAndetBillede.png";  
Pic.alt = "dette er en alt tekst";
```

Alternativt:

```
pic.setAttribute("src", "etAndetBillede.png");
```

OBS!

Enhver attribut kan ændres
på denne måde!

<https://developer.mozilla.org/en-US/docs/Web/API/Element/setAttribute>

Nyt indhold med createElement og appendChild

```
let info = document.querySelector("#info");  
info.innerHTML="<h1>Min overskrift til info</h1><img src='http://mabe-kea.dk/E19/pics/pig.png'>"
```

Alternativ:

```
let h1 = document.createElement("h1");  
let overskrift = document.createTextNode("Min overskrift til info");  
h1.appendChild(overskrift);  
let img = document.createElement("img");  
img.src="http://mabe-kea.dk/E19/pics/pig.png"  
info.appendChild(h1);  
info.appendChild(img);
```

Resultat:

```
<div id="info">  
  <h1>Min overskrift til info</h1>  
    
</div>
```

HTML DOM createElement() Method, : https://www.w3schools.com/jsref/met_document_createelement.asp

HTML DOM appendChild() Method, : https://www.w3schools.com/jsref/met_node_appendchild.asp

textContent vs innerHTML vs createElement/appendChild

- Brug **textContent** til rene tekster
- Brug **innerHTML** til at erstatte/indsætte html
- Brug **createElement** og **appendChild** til html-tilføjelser i en mere struktureret form

Overvejelser ved brug af innerHTML:

<https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML>

DOM-manipulationer - endnu flere

```
const info = document.querySelector("#info");
```

```
info.textContent = ... // indsætter tekst i info  
info.textContent += ... // tilføjer tekst til info
```

```
info.setAttribute('attributNavn', 'værdi')  
// sætter en attribut-værdi på info  
info.getAttribute('attributNavn')  
// læser en værdi fra en attribut i info  
info.removeAttribute('attributNavn')  
// fjerner en attribut
```

```
info.src = ... // sætter src-værdi (img-tag)  
info.alt = ... // sætter img's alt-værdi  
info.href = ... // sætter href-værdi (a-tag)
```

```
info.innerHTML = // indsæt html-kode i info  
info.innerHTML += ... // tilføjer html til info  
info.insertAdjacentHTML(position, html-kode)  
// indsætter indhold på en bestemt placering i forhold  
til info
```

Ref: https://www.w3schools.com/jsref/met_node_insertadjacenthtml.asp

```
info.cloneNode(true) // kopierer et element  
let element = document.createElement("img");  
info.appendChild(element) // tilføjer et element  
info.removeChild(element) // fjerner et element
```

Begreber

`querySelector` - syntaksen for at vælge et DOM element

`DOM selectors` - identifikationen af det enkelte DOM element i en `querySelector` Sætning

`textContent`

`innerHTML`

`createElement > appendChild`

Øvelse 2A - Udvælg DOM-elementer

1. Lav en ny Branch
2. Opret en js fil, og lav querySelectors på 5 forskellige elementer fra øvelse 1, og vis dem med console.log()
3. Eksperimentér med **begge** nedenstående metoder. Vælg den du bedst kan lide!
4. Commit, Merge og Push til GitHub

Tips!

Du får måske brug for en DOM-selector til et elements nærmeste søskende:

https://developer.mozilla.org/en-US/docs/Web/CSS/Adjacent_sibling_combinator

ELLER

Du kan bruge en pseudo selector:

<https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child>

Øvelse 2B - Udskift tekstindhold i elementer

1. Lav en ny Branch
2. Udskift teksten i titlen v.h.af. JavaScript
3. Udskift tekst og overskrifter i de to article elementer v.h.af. JavaScript
4. Commit, Merge og Push til GitHub

Udfordring!

Brug `textContent` til teksten i den ene article, og `innerHTML` med f.eks `` tag i den anden.
Eksperimenter og se forskellen!

Øvelse 3 - Udskift billeder

1. Lav en ny Branch

Brug javascript til følgende:

2. Udskift billederne i de to article tags
3. Udskift eller tilføj tekst i alt attributten på billederne.
4. Commit, Merge og Push til GitHub

Øvelse 4 - Tilføj nyt element

Lav en ny Branch.

Tilføj nyt javascript som:

1. opretter et nyt article-element med(createElement).
2. tilføjer et billede, en overskrift og noget tekst til den nye article.
3. Indsætter det nye element efter de eksisterende article elementer (appendChild)
4. Commit, Merge og Push til GitHub

EventListener og eventhandler

EventListener & EventHandler

```
let element = document.querySelector("#info");
```

```
element.addEventListener("click", doSomething);
```

```
function doSomething() {  
    // det der skal ske  
}
```

Event - hvad skal der ske?

EventListener

Kald til EventHandlerler-funktion

EventHandler
- Den funktion der bliver
udført når Event indtræffer

For at fjerne eventlistener fra element igen:

```
element.removeEventListener("click", doSomething); // holder op med at virke
```

ContentLoaded - eventlistener og -handler

Javascript bør først udføres, når man er helt sikker på, at DOM'en er loadet



JavaScript HTML DOM EventListener, W3Schools: https://www.w3schools.com/js/js_htmlDOM_eventlistener.asp
DOMContentLoaded, MDN Webdocs: <https://developer.mozilla.org/en-US/docs/Web/Events/DOMContentLoaded>

Flere events

```
let info = document.querySelector("#info");  
  
info.addEventListener("click", doSomething);  
info.addEventListener("dblclick", doSomething);  
info.addEventListener("mouseover", doSomething);  
info.addEventListener("mouseout", doSomething);
```

```
function doSomething() {  
    // det der skal ske  
}
```

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

https://www.w3schools.com/jsref/dom_obj_event.asp

Endnu flere:

mousedown
mousemove
mouseup
touchdown
touchmove
touchend
animationend
transitionend
keydown
keypress
keyup

...

3 måder at kalde funktioner på i eventListeners

```
let element = document.querySelector("#info");
```

1. Almindelig funktionskald:

```
element.addEventListener("click", doSomething);  
function doSomething() {  
    // det der skal ske  
}
```

2. Anonym funktion:

```
element.addEventListener("click", function() {  
    // det der skal ske  
});
```

3. Arrow-funktion:

```
element.addEventListener("click", () =>{  
    // det der skal ske  
});
```

Begreber

Event

EventListener

EventHandler

Anonym funktion

Arrow-funktion

Øvelse 5 - test om DOM'en er loadet

- Ny Branch
- Arbejd videre på indholdet:
Du skal sørge for at scriptet tester, om DOM'en er loadet, inden der bliver ændret på indholdet.
- Commit - Merge - Push

Øvelse 6 - click event

1. Ny Branch
2. Arbejd videre på indholdet:
Når der klikkes på det første billede, skal billedet skiftes ud.
3. Merge, Commit og Push til GitHub

TIP!

Hver gang vi kalder url'en fra `picsum.photos`, får vi et nyt tilfældigt billede. Ex.:

```
img1.src = "https://picsum.photos/400/300";
```