Javascript
DOM

Bjarte Kileng

HVL

2. november, 2021

DOM (Document Object Model)

- DOM lar oss arbeide med HTML, XML og CSS dokumenter.
- Inkluderer metoder og egenskaper for å referer, lage og modifisere elementer på websiden.
- For aktuelle relevante standarder og dokumenter, se e.g.:
  - Web APIs|MDN.
  - HTML Living Standard (HTML 5 og DOM 2)
  - DOM Living Standard (erstatter DOM 3 og DOM 4)
- I dette kurset skal vi se på hvoran vi refererer elementer.
- JavaScript for å modifisere dokumenter kommer i DAT152.
  - I DAT108 vil vi kun gjøre enkle modifikasjoner.

Lister av web-elementer

- Flere DOM metoder returnerer lister med elementer, f.eks. *getElementsByTagName()*.
- Listene er valigvis «levende».
  - Kun *querySelectorAll()* returnerer en statisk liste.
- En levende liste oppdateres automatisk når dokumentstrukturen endres.
  - Listen vil alltid reflektere websiden slik den er akkurat nå, selv om listen ble opprettet tidligere.
- Listene kan bare oppdateres ved å endre på web-dokumentet.

Elementlister i *Prototype* og *jQuery*

- Rammeverkene *Prototype* og *jQuery* inkluderer metoder som returnerer lister av web-elementer.
- Disse listene implementeres som forekomster av JavaScript **Array**.
- Disse listene er statiske. Endringer i dokumentet vil ikke forplante seg til eksisterende lister.

Absolutte og relative dokument-referanser

- Absolutte referanser tar utgangspunkt i *document*.

```
const root = document.getElementById("root");
```

- Relative referanser tar utgangspunkt i et annet web-element.

```
const submitButton = root.querySelector("button[submit]");
```

Bruk av absolutte dokument-referanser

- Absolutte referanser gjør at koden ikke er portabel.
- Brukes i controller kode, f.eks. callback ved **DOMContentLoaded**.
  - F.eks. referere et beholder-element i HTML-dokumentet.
- Eksempel med absolutt referanse:
  - HTML:

```
<div id="root">
    <!-- Mer HTML her -->
</div>
```

  - JavaScript:

```
function init() {
    const rootElm = document.getElementById("root");
    const gui = new DemoGUI(rootElm);
}
```

Lage absolutt referanse

- Vi vil kun trenge metoden `document.getElementById`.
- Metoder for relative referanser kan også brukes på *document*.
  - Da lager de absolutte referanser.
- Finnes diverse *document* egenskaper som er absolutte referanser.
  - F.eks. *document.body* og *document.documentElement*.

Bruk av relative dokument-referanser

- Adresseres med utgangspunkt i et annet HTML-element.
  - Vanligvis i trestruktrukturen under et beholder-element.
  - Metoder/egenskaper også for å finne foreldre og søsken.
- Bortsett fra i kontroller kode bør alle referanser være relative.
- Kan finne element utifra tagg, foreldre/søsken/barn relasjoner, *class* attributtet og andre HTML-attributter.
- HTML 5 tillater egendefinerte attributter som starter med «data-».

```
<table>
    <tr><th>Navn</th><th>Telefonnummer</th></tr>
    <tr data-StudentID="5"><td>Ole</td><td>12345678</td></tr>
    <tr data-StudentID="9"><td>Ane</td><td>87654321</td></tr>
</table>
```

Lage relative referanser

- Finne element utifra egenskap ved element:
  - F.eks. *querySelector()*, *getElementsByTagName()*, *getElementsByClassName()*, *querySelectorAll()*.
- Mange elementer har egne tilpassede egenskaper.
  - HTML *table*, *thead* og *tbody* har egenskap *rows*.
  - HTML tabellrekker, tagg *tr* har egenskap *cells*.
- Finne element utifra relasjon:
  - F.eks. *parentNode*, *children*, *firstElementChild*, *lastElementChild*, *previousElementSibling*, *nextElementSibling*.

getElementsByTagName()

- Returnerer en liste av HTML-elementer med gitt tagg.
- Argument «*» gir en liste av alle elementer som har tagg.
  - Tekst og kommentarer er også elementer, men uten tagg.
- Finne alle elementer i strukturen under *elmRef* med tagg *table*:

```
const tableElements = elmRef.getElementsByTagName("table");
```

- Finne alle elementer i strukturen under *elmRef* med vilkårlig tagg:

```
const tableElements = elmRef.getElementsByTagName("*");
```

- Finne første element under *elmRef* med tagg *button*:

```
const buttonRef = elmRef.getElementsByTagName("button")[0];
```

getElementsByClassName()

- Returnerer en liste av HTML-elementer med gitt *class* attributt.

```
<div id="root">
    <p class="firstname">Ole</p>
    <p class="firstname">Anne</p>
</div>
```

- Liste av alle elementer under *elmRef* med *class* attributt **firstname**:

```
const elements = elmRef.getElementsByClassName("firstname");
```

- Finne andre element under *elmRef* med *class* attributt **firstname**:

```
const elements = elmRef.getElementsByClassName("firstname")[1];
```

- Antall elementer under *elmRef* med *class* attributt **firstname**:

```
const count = elmRef.getElementsByClassName("firstname").length;
```

querySelector()

- Bruker CSS-selektor for å finne element.
- Returnerer første element som stemmer med selektor, eller **null**.
- Finne første tabellrekke under *elmRef* med attributt *data-userID*.

```
const elm = elmRef.querySelector("tr[data-userID]");
```

- Finne første tabellrekke under *elmRef* med attributt *data-userID="5"*:

```
const elm = elmRef.querySelector('tr[data-userID="5"]');
```

- Første element uansett tagg med attributt *data-userID="5"*:

```
const elm = elmRef.querySelector('[data-userID="5"]');
```

- Siste tabellcelle i rekke med attributt *data-userID*:

```
const elm = elmRef.querySelector('tr[data-userID]>td:last-child);
```

querySelectorAll()

- Returnerer liste av alle elementer som stemmer med selektor,
- Returnerer en statisk liste, treff på selektor da metoden ble kjørt.
- Alle tabellceller i *tbody* som er første barn:

```
const rowElements = elmRef.querySelectorAll("tbody td:first-child");
```

- Alle tabellceller som er barn to av rekke med attributt *data-userID*:

```
const selector = 'tr[data-userID]>td:first-child + td';
const cellElements = elmRef.querySelectorAll(selector);
```

Dokument-referanser som element egenskaper

- Elementer kan ha egenskaper og metoder tilpasset til elementet.
  - Kan alltid bruke de generelle metodene, men
  - de tilpassede metodene kan gi enklere kode.
- HTML *table* har egenskaper *tHead*, *rows* og *tBodies*.
  - *tHead* referer tabellen sin *thead*.
  - *tBodies* er en levende liste av alle *tbody* elementer.
  - *rows* er en levende liste av alle tabellrekkene i tabellen.
- HTML *tbody* og *thead* har egenskap *rows*.
  - Levende liste av alle tabellrekkene.
- Tabellrekker, tagg *tr* har egenskap *cells*
  - Levende liste av alle tabellcellene.
- HTML *select* har en egenskap *options*.
  - Levende liste av alle options-elementene.
- Dokumentasjonen for det enkelte element gir flere eksempler.

Egenskaper for element-relasjoner

```
<form action="behandledate" method="POST">
    Fyll inn navn: <input type="text" />
    <button type="submit">Send</button>
    <button type="reset">Nullstill</button>
</form>
```

- Egenskapen *childred* er en levende liste av alle barn.
  - Form-elementet har tre *children*, et *input* og to *button* elementer.
- *firstElementChild* er første barn i listen av barneelementer.
  - Form-elementet sitt *firstElementChild* er elementet *input*.
- *lastElementChild* er siste barn i listen av barneelementer.
  - Form-elementet sitt *lastElementChild* er reset-knappen.
- *previousElementSibling* er søskenelementet foran.
  - Reset-knappen sin *previousElementSibling* er submit-knappen.
- *nextElementSibling* er søskenelementet etter.
  - Submit-knappen sin *nextElementSibling* er reset-knappen.
- *parentNode* er elementet sitt forelder element.

Modifisere dokument

- DOM inkluderer metoder og egenskaper for å modifisere dokumentet.
- *innerText* og *textContent* er all teksten i dokumentstrukturen under elementet.
- *innerHTML*, *outerHTML* og *insertAdjacentHTML* lar oss arbeide med dokumentstrukturen.
- *classList* lar oss arbeide med elementet sitt HTML attributt *class*.

```
<span class="fornavn student">Ole</span>
```

  - Eksempelet har to verdier i sitt *class* attributt: *fornavn* og *student*.

innerText og *textContent*

- *innerText* returnerer synlig tekst i element.
- *textContent* returnerer all tekst i element.
- Eksempel der *innerText* og *textContent* gir ulike svar:

```
<p>Hei <span style="display:none">dere</span></p>
```

- Kan både lese og endre tekstinnhold.

```
elmRef.innerText = "Adjø";
```

- Kun ren tekst, ikke HTML kan legges til element.
  - Tilsvarende egenskaper for HTML kan gi sikkerhetshull.
  - Forsiktig med *innerHTML*, *outerHTML* og *insertAdjacentHTML*.

classList

- Levende liste av alle *class* objekter til HTML elementet.
- Kan endre utseende ved å modifisere elementet sitt *class* attributt.
- Metode *add* legger til et nytt objekt til elementet sitt *class* attributt.
- Metode *remove* fjerner et objekt fra elementet sitt *class* attributt.
- Egenskap *length* er antall objekter i elementet sitt *class* attributt.
- Metode *contains* sjekker for objekt i elementet sitt *class* attributt.
- Metode *toggle* «toggler» objekt i elementet sitt *class* attributt.
  - Returnerer **sann** hvis objekt nå finnes i *class* attributtet.
- Metode *replace* erstatter et objekt med et annet.
- Metode *values* returnerer iterator over alle objekter i *class* attributtet.

innerHTML, outerHTML og insertAdjacentHTML

- Disse metodene lar oss enkelt opprette nye HTML-strukturer.
- Disse metodene må **aldri** brukes på brukerdata.
- Eksempelet under viser riktig bruk av *insertAdjacentHTML*:

```
showMessage(message) {
    const elmRef = this.rootElement.querySlector("div[data-messages]");
    elmRef.insertAdjacentHTML('beforeend', "<p></p>");
    elmRef.lastElementChild.textContent = message;
  }
```

- Følgende må aldri gjøres:

```
showMessage(message) {
    const elmRef = this.rootElement.querySlector("div[data-messages]");
    elmRef.insertAdjacentHTML('beforeend', `<p>${message}</p>`");
  }
```

- Koden i siste eksempel gjør applikasjonen sårbar for XSS angrep.
  - *innerHTML* og *outerHTML* er akkurat like problematiske.
  - Koden er sårbar for injeksjon av HTML og CSS, men ikke JavaScript.
- *insertAdjacentHTML* kan være mye raskere enn *innerHTML*.