Web Development

JavaScript – Labo 6

Ophalen van DOM-tree elementen

- document.**getElementById**(id)
- document.**getElement<u>s</u>ByClassName**(className)
- document.**getElement<u>s</u>ByTagName**(tagName)

- document.**querySelector**(selector)
- document.**querySelector<u>All</u>**(selector)

Selectors

```html
<body>
 <h1>Hoofding</h1>
 <p>Par. 1</p>
 <p class="a">Par. 2</p>
 <p id="B">Par. 3</p>
 <p class="a">Par. 4</p>
</body>
```

```
document.querySelector('#B')
document.querySelectorAll('p')
document.querySelector('.a')
document.querySelector('body > p')
```

Zoeken naar kinderen

```
let main = document.getElementsByTagName('main')[0];

let mainPs = main.getElementsByTagName('p');
let mainBolds = main.querySelectorAll('.bold');

// OF
let mainPs = document.querySelectorAll('main p');
let mainBolds =
        document.querySelectorAll('main .bold');
```

HTML aanpassen

- .innerHTML
  - Voordeel: gemakkelijk en alles in één keer
  - Nadelen
    - Veel werk voor de browser (parsen van HTML, verwijderen van bestaande elementen, lay-out opnieuw maken)
    - Geen verwijzing naar de nieuwe elementen (event listeners kunnen we pas koppelen na opvragen van de nieuwe DOM-tree elementen)
    - Enkel kinderen vervangen

DOM-tree nodes

```
main.parentNode
// één node, <body>
```

```
<body>
  <main>
    <!-- main content -->
    <p>1</p>
    <p>2</p>
  </main>
</body>
```

DOM-tree nodes

```
main.firstChild
// eerste child node
main.lastChild
// laatste child node
main.firstElementChild
// eerste child element
main.lastElementChild
// laatste child element
```

```
<body>
  <main>
    <!-- main content -->
    <p>1</p>
    <p>2</p>
  </main>
</body>
```

DOM-tree nodes

```
p1.nextSibling
// volgend broertje/zusje
p1.previousSibling
// vorig broertje/zusje
p1.nextElementSibling
// volgend broertje/zusje (element)
p1.previousElementSibling
// vorig broertje/zusje (element)
```

```
<body>
  <main>
    <!-- main content -->
    <p>1</p>
    <p>2</p>
  </main>
</body>
```

DOM-tree nodes

```
main.childNodes
// lijst met nodes
main.children
// lijst met elementen
```

```
<body>
  <main>
    <!-- main content -->
    <p>1</p>
    <p>2</p>
  </main>
</body>
```

Node properties

```
p1.nodeName

p1.nodeType

p1.childNodes[0].nodeValue
```

- De naam van de node (‘p’) OF *#text*
- Getal dat overeenkomt met het type node (1 voor elementen, 3 voor tekst, 8 voor comments…)
- De tekstwaarde van de node (enkel bij #text)

Nodes aanmaken

- Element nodes aanmaken

```
let newP = document.createElement('p');
```

- Tekst nodes aanmaken

```
let newText = document.createTextNode('Tekst.');
```

Attributen

```
let source = img.getAttribute('src');

img.setAttribute('src', 'img/afb1.png');

img.hasAttribute('alt');
```

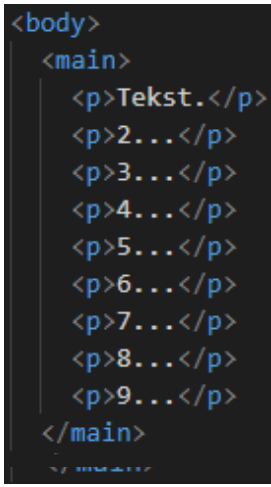
Data-attributen

- “Custom” attributen
- Kunnen gebruikt worden in eigen applicatie
- Blijft geldige HTML (ook in de validator)

```
<span data-imdb="tt0112281">Ace Ventura</span>
```

Bye bye innerHTML

```
let main = document.getElementsByTagName('main')[0];
let oldP = main.firstElementChild;
main.removeChild(oldP);
let newP = document.createElement('p');
let newText = document.createTextNode('Tekst.');
newP.appendChild(newText);
main.prepend(newP);
```



Event handling

```
let foto1 = document.querySelector('#foto1');
let foto2 = document.querySelector('#foto2');

foto1.addEventListener('click', gekliktFoto1);
foto2.addEventListener('click', gekliktFoto2);
```

Wat met 100 foto’s?

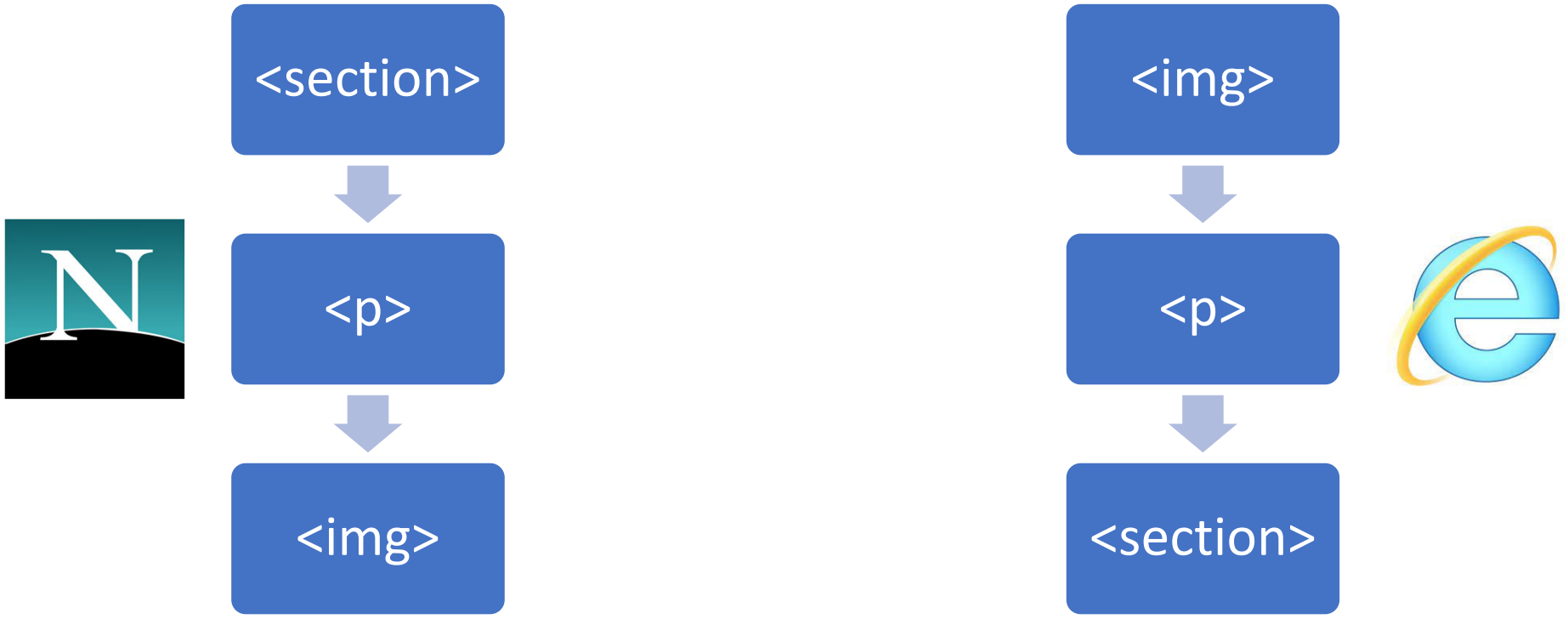
Events uitvoeren

```html
<section>
  <p>
    <img src="..." alt="..." />
  </p>
</section>
```

```js
let section = document.querySelector('section');
let p = document.querySelector('p');
let img = document.querySelector('img');

section.addEventListener('click', geklikt);
p.addEventListener('click', geklikt);
img.addEventListener('click', geklikt);
```

Capturing & bubbling



Capturing & bubbling

- W3C: gulden middenweg
- Eerst capturing tot het element bereikt is, dan bubbling naar boven
- Extra parameter *capture*
  - True: capturing
  - False: bubbling (default)

```
section.addEventListener('click', geklikt, true);
p.addEventListener('click', geklikt, false);
img.addEventListener('click', geklikt);
```

Events

- Hoe weten we nu welk event aan het uitvoeren is?
- Kunnen een parameter binnenkrijgen die het event voorstelt

```
const geklikt = (event) => { /*...*/ }
```

  - event.**target**
    - Het element waar het event zich voordoet
  - event.**currentTarget**
    - Het element wiens event listener wordt uitgevoerd

Voorbeeldje

```
const geklikt = (event) => {
  console.log(event.target.name);
  console.log(event.currentTarget.name);
}
```

Andere events stoppen

- event.**stropPropagation**()
  - Niet verder “bubbelen” of “capturen”
- event.**preventDefault**()
  - Standaardgedrag onderdrukken
  - Bijvoorbeeld bij een submit-knop zorgen dat het formulier niet verzonden wordt

- Combinatie van beide?
  - *return false* in event listener

Labo