JavaScript Deel 4

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

DOM Inleiding

Wat is het DOM?

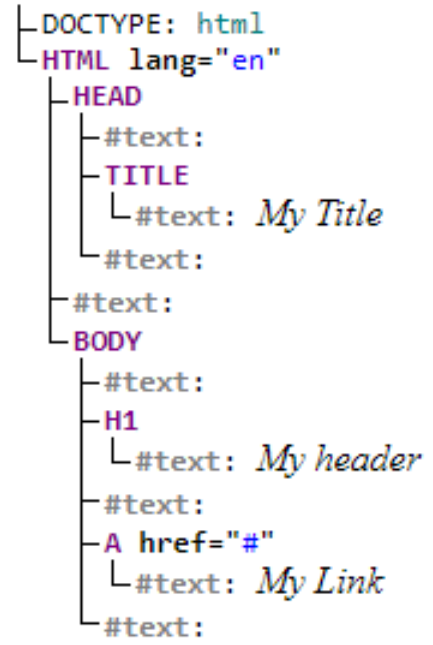
- World Wide Web Consortium Document Object Model (W3C DOM)
- Model van de structuur van de webpagina, wordt gebruikt om de pagina op het scherm (of ander medium) weer te geven.
- ‘Levende’ datastructuur dat we via een script kunnen manipuleren.

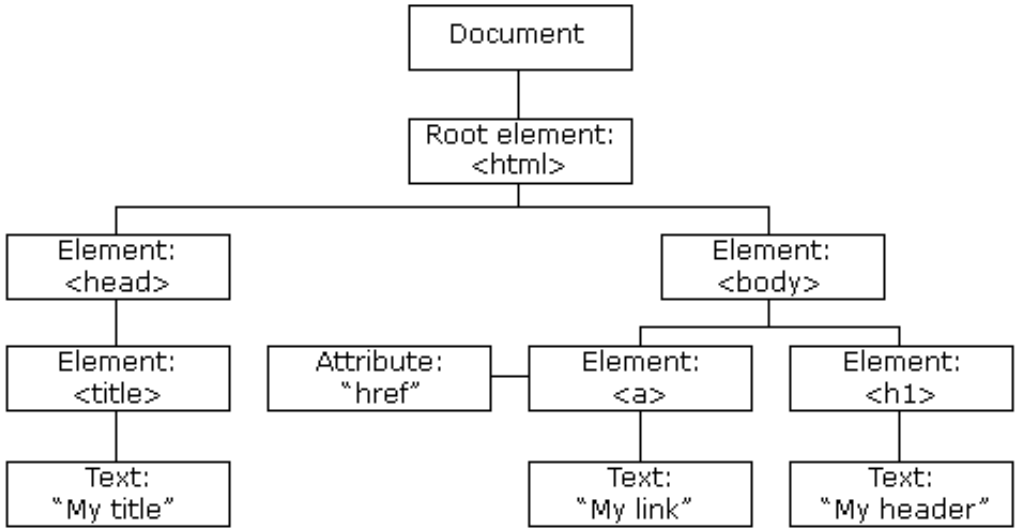
Wat is het HTML-DOM

- Het standaardmodel voor HTML-documenten.

- Het bevat:
  - Alle HTML-elementen als objecten
  - De attributen of eigenschappen van alle HTML-elementen
  - De methoden om toegang te krijgen tot alle HTML-elementen
  - De events voor alle HTML-elementen

```
<!DOCTYPE html>
<html lang="en">
<head>
	<title>My Title</title>
</head>
<body>
	<h1>My header</h1>
	<a href="#">My Link</a>
</body>
</html>
```





Relatie HTML DOM en Javascript

- Via DOM manipulatie kunnen we dynamische webpagina's maken:
  - HTML-elementen benaderen, aanpassen, verwijderen, toevoegen
  - HTML-attributen aanpassen, verwijderen, toevoegen
  - CSS-eigenschappen aanpassen
  - Reageren op events
  - Nieuwe events creëren
  - …

Belangrijke begrippen

- Vooraleer we aan de slag gaan met het HTML DOM, moeten we eerst enkele belangrijke begrippen onder de knie krijgen!

- Alles in een HTML document noemen we een *'node'* en alle nodes kunnen we aanspreken en gebruiken in Javascript. We kunnen nieuwe nodes creëren en bestaande nodes aanpassen of verwijderen.

Belangrijke begrippen

Document node	Het volledige document
Element node	Een HTML-element
Text node	De tekst in een HTML-element
Attribute node	Het attribuut van een HTML-element
Comment node	Een stukje commentaar

Belangrijke begrippen

- Alle nodes hebben een hiërarchische relatie tegenover elkaar.

Root node	De top node, de hoogste node in een document.
Parent node	Elke node heeft exact één parent node, met uitzondering van de root node.
Child node	Een parent node kan meerdere child nodes hebben.
Siblings	Child nodes met dezelfde parent node zijn siblings.

DOM manipulatie – Deel 1

Benaderen van element nodes

- Er bestaan verschillende HTML DOM methoden om bestaande HTML-elementen te zoeken en te benaderen in het DOM:
  - Via id
  - Via tag-naam
  - Via class-naam
  - Via naam
  - Via CSS-selector

- Het resultaat van deze methoden is ofwel één object, ofwel een verzameling objecten.

Benaderen via id

- We kunnen via de methode getElementById() één uniek HTML-element benaderen in het DOM door gebruik te maken van de id van dat element.

```
// Geeft het element als één object terug.
let object = document.getElementById(“demo”);
```

Benaderen via tag-naam

- We kunnen via de methode getElement**s**ByTagName() alle HTML-elementen van eenzelfde type benaderen in het DOM door gebruik te maken van de tag-naam.

```
// Geeft alle p-elementen als een object collectie terug.
let object = document.getElements**s**ByTagName(“p”);

// Geeft het eerste p-element als een object terug.
let object = document.getElements**s**ByTagName(“p”)[0];
```

Benaderen via class-naam

- We kunnen via de methode getElement**s**ByClassName() meerdere HTML-elementen met dezelfde class benaderen in het DOM.

```
// Geeft alle elementen met de class ‘demo’ als een object collectie terug.
let object = document.getElements**s**ByClassName(“demo”);
```

Benaderen via name

- We kunnen via de methode getElement**s**ByName() meerdere HTML-elementen met dezelfde name benaderen in het DOM.

```
// Geeft alle elementen met de name 'volledigenaam' als een object collectie
   terug.
let object = document.getElements**s**ByName("volledigenaam");
```

Benaderen via CSS-selector

- We kunnen via de methode querySelectorAll() meerdere HTML-elementen die met een specifieke CSS-selector (id, class, tag, geneste selector, …) matchen, benaderen in het DOM.

```
// Geeft het element met de CSS-selector “p.demo” als een object terug.
let object = document.querySelectorAll(“p.demo”);
```

Benaderen via CSS-selector

- Verschil tussen methode **querySelectorAll()** en **querySelector()**:
  - De methode querySelectorAll() benadert alle elementen die met de opgegeven CSS-selector matcht.
  - De methode querySelector() benadert slechts één element die met de CSS-selector matcht. Matchen er meerdere elementen, wordt het eerste element als object teruggegeven.

DOM manipulatie – Deel 2

Inhoud van HTML-elementen aanpassen

- De makkelijkste manier om de inhoud van een html-element aan te passen is de eigenschap ‘innerHTML’.

```
// De paragraaf met id ‘uitkomst’ krijgt een nieuw stukje tekst.
document.getElementById("uitkomst").innerHTML = "Nieuwe tekst!";
```

Attribuut-waarden aanpassen en toevoegen

- Via de attribuutnaam kunnen we de waarden van een HTML-attribuut aanpassen of een HTML-attribuut toevoegen.

```
// De afbeelding met id ‘logo’ krijgt een nieuwe image source.
document.getElementById(“logo").src = "assets/logo2.png”;
```

CSS-eigenschappen aanpassen en toevoegen

- Via de stijleigenschap kunnen we de opmaak van specifieke HTML-elementen aanpassen.

```
// De paragraaf met id ‘uitkomst’ krijgt de tekstkleur rood.
document.getElementById("uitkomst").style.color = "#f00";
```

Element nodes en text nodes toevoegen

- We kunnen via Javascript ‘runtime’ nieuwe HTML-elementen met inhoud aanmaken. Dit doen we met de volgende twee functies:
  - Met ***‘createElement()’*** maken we het nieuwe HTML-element aan.
  - Met ***‘createTextNode()’*** maken we de platte tekst die als inhoud in het HTML-element moet komen te staan.

Element nodes en text nodes toevoegen

- Naast het aanmaken van de HTML-elementen met inhoud, moeten we ook aangeven waar deze elementen mogen worden toegevoegd aan het DOM:
  - Met ***'appendChild()'*** kan je een node toevoegen als laatste node aan een geselecteerd element.
  - Met ***'insertBefore()'*** kan je een node toevoegen aan een geselecteerd element en aangeven op welke locatie binnen dat geselecteerde element.

Voorbeeld met appendChild()

```
▼<header id="startheader">
    <h1>Welkom op deze website!</h1>
    <p id="test">Dit is een stukje tekst</p>
  </header>
```

```
// Maak een nieuwe paragraaf aan.
let nieuweParagraaf = document.createElement("p");

// Geef de paragraaf optioneel een id (bijvoorbeeld 'test').
nieuweParagraaf.id = "test";

// Maak een textnode aan.
let tekstParagraaf = document.createTextNode("Dit is een stukje tekst");

// Voeg de textnode toe aan de nieuwe paragraaf.
nieuweParagraaf.appendChild(tekstParagraaf);

// Voeg de paragraaf toe aan de header als laatste child-element.
document.getElementById("startheader").appendChild(nieuweParagraaf);
```

Voorbeeld met insertBefore()

```
▼<header id="startheader">
    <p id="test">Dit is een stukje tekst</p>
    <h1 id="titel">Welkom op deze website!</h1>
  </header>
```

```
// Maak een nieuwe paragraaf aan.
let nieuweParagraaf = document.createElement("p");

// Geef de paragraaf optioneel een id.
nieuweParagraaf.id = "test";

// Maak een textnode aan.
let tekstParagraaf = document.createTextNode("Dit is een stukje tekst");

// Voeg de textnode toe aan de nieuwe paragraaf.
nieuweParagraaf.appendChild(tekstParagraaf);

// Voeg de paragraaf toe aan de header als eerste child-element.
document.getElementById("startheader").insertBefore(nieuweParagraaf,
document.getElementById("titel"));
```

In te voegen element

Voorgaand element

Verwijderen van bestaande HTML-elementen

- Met ‘removechild()’ kunnen we een element uit het DOM verwijderen.

```
// Verwijder de laatste paragraaf in de header uit het DOM
let header = document.getElementById(“startheader”);
let paragraaf = header.querySelectorAll(“p:last-child”)[0];
header.removeChild(paragraaf);
```

Overige functies en eigenschappen

- Javascript heeft nog veel meer functies en eigenschappen om met het DOM aan de slag te gaan. Bekijk dus zeker ook eens deze informatie op W3Schools!
  - https://www.w3schools.com/js/js_htmldom_navigation.asp
  - https://www.w3schools.com/js/js_htmldom_nodes.asp