Javascript – deel 02

Deze les behandelt de control flow mogelijkheden van javascript, eenvoudige input en output en hoe je teksten kunt opvragen en aanpassen in de DOM-tree.

Permanente evaluatie

In de volgende les kan je docent je oplossingen van de opdrachten opvragen en laten meetellen voor je permanente evaluatie.

Verslag

In dit deel van de cursus staan verschillende vragen die je moet beantwoorden en opdrachten om iets te maken of uit te proberen. Het is belangrijk dat je alle opdrachten zorgvuldig uitvoert!

Documenteer je werk in een verslag document 'javascript deel 02' waarin je

- voor elke uitprobeer opdracht een entry maakt met screenshots ter staving van wat je deed
    - (de codecademy delen hoef je niet te documenteren)
- je antwoorden op de gestelde vragen neerschrijft

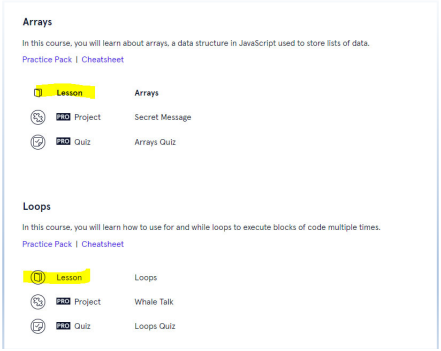
Oplossingen van grotere opdrachten (met veel code) bewaar je aparte folders in een Webstorm project.

Loops and Arrays

Ga naar de "Introduction to Javascript" cursus op Codecademy

https://www.codecademy.com/learn/introduction-to-javascript

En doorloop de volgende secties:



Arrays

Een array is een lijst van elementen die ingesteld en opgevraagd kunnen worden op basis van hun positie.

In veel programmeertalen is een array een primitief datatype met een vaste grootte. Er is een beperkt aantal slots waarin elementen kunnen geplaatst worden en dit aantal moet op voorhand worden opgegeven. Om elementen in te lassen of te verwijderen of het aantal beschikbare slots te wijzigen, moet er dikwijls omslachtige code geschreven worden die elementen kopieert.

In javascript zijn arrays echter veelzijdiger en lijken qua mogelijkheden veel meer op bv. een ArrayList uit Java. Er zijn methoden voorhanden om makkelijk inlassingen, toevoegingen of verwijderingen te realiseren en hun grootte (het aantal slots) kan variëren.

Javascript gebruikt net als veel andere programmeertalen de rechte haakjes [ ] voor arrays. Dit maakt het natuurlijk verwarrend omdat ze eruit zien als een Java array, maar zich gedragen als een ArrayList.

Je kunt als volgt een leeg array aanmaken :

```
let leeg = [];
```

Een voorgevuld array aanmaken

```
let teksten = ["een","twee","drie"];
let getallen = [1, 2, 3];
let gemixt = [true, "hallo", 4];                    // verschillende soorten elementen!
```

Let erop dat een array in javascript waarden mag bevatten van verschillende types, zoals het 'gemixt' voorbeeld hierboven demonstreert.

Het aantal elementen in een array opvragen gebeurt via de .length property :

```
let elementen = ["een", "twee", "drie"];
    console.log ( elementen.length );          // toont 3
```

Elementen in een array worden op basis van hun posities aangeduid, beginnend bij 0. Het eerste element staat dus op indexpositie 0.

Je kan Individuele element opvragen op basis van een index :

```
let elementen = ["een", "twee", "drie"];
console.log( elementen[0] );          // toont "een"
console.log( elementen[1] );          // toont "twee"
console.log( elementen[2] );          // toont "drie"
```

Om alle elementen in een array te overlopen kunnen we een for-loop gebruiken

```
let elementen = ["een", "twee", "drie"];

for (let i=0 ; i<elementen.length ;
        i++) { console.log(
        elementen[i] );
}
```

Enkele andere nuttige methods om makkelijker met arrays te werken :

indexOf(element, idx)
lastIndexOf(element, idx)

geeft de laagste/hoogste index waarop het element gevonden
werd elementen worden vergeleken met ===
indien het elementen niet wordt gevonden, geeft dit -1 terug
de idx parameter is optioneel en geeft aan op welke index de zoektocht moet beginnen

push(element)

voegt het element toe op het einde van het array
geeft de nieuwe lengte terug

pop()

verwijdert het laatste element van het array
geeft het verwijderde element terug

unshift(element)

voegt het element toe aan het begin van het array geeft de nieuwe lengte terug

shift()

verwijdert het eerste element van het array geeft het verwijderde element terug

slice(i1, i2)

geeft een nieuw array terug met de elementen vanop index i1 tot index i2 (exclusief)

join() de methode Array.join() converteert alle elementen in array naar een string en voegt deze samen tot één lange string. Deze string wordt geretourneerd. Optioneel kan je een scheidsteken meegeven. Als je niets meegeeft, worden de array-elementen in de retourneerde string met een komma van elk gescheiden

Opdracht: Arrays

Maak een array met namen van familieleden aan. Zorg dat er minimaal vijf zijn. Voer daarmee volgende opdrachten uit:

- schrijf naar de console hoeveel elementen de array bevat
- schrijf het eerste, derde en vijfde element uit de array naar de console
- Vraag met prompt() een extra naam op en voeg deze toe aan de Array. Probeer dit via een zelf geschreven functie VoegNaamToe. Maak gebruik van pass-by-reference. Schrijf vervolgens het resultaat naar de console.
- Converteer de array naar een string en toon deze op de console.

Eenvoudige input en output

In elke browser heb je een console venster dat je als programmeur kunt gebruiken voor technische meldingen. Als gewone gebruiker heb je daar geen boodschap aan en krijg je dit venster dus ook niet te zien.

In Chrome kun je het console venster tevoorschijn halen via de Developer tools (tabblad Console).

Je kunt in je javascript programma, tekst op de console plaatsen met de volgende opdracht :

```
console.log("dit is een mededeling op de console");
```

Op de console kun je trouwens ook opdrachtjes intypen en uitvoeren, handig om snel iets uit te proberen.

In javascript kun je drie soorten popups op het scherm plaatsen :

```
window.alert("Dit is een mededeling");
```

Toont de mededeling in een popup met een 'ok' button

window.confirm("Weet u het zeker?");

Toont de vraag in een popup met een 'ok' en 'cancel' button

De return waarde geeft aan op welke knop de gebruiker duwde.

window.prompt("Wat is uw naam", "onbekend");

Toont de vraag in een popup met een tekstveld met een 'ok' en een 'cancel' button.

De andere tekst ("onbekend" hierboven) wordt standaard al ingevuld bij het tonen.

De console en window popups zijn geschikt voor ontwikkelaars, niet voor eindgebruikers. Als je programma output voor de eindgebruiker heeft, zul je deze via de DOM-tree moeten tonen.

Opdracht 01

Probeer eerst eens elk van de drie soorten popups uit in een javascript programma.

Zet daarna de return value van de confirm call op de console en probeer uit met zowel op 'ok' als op 'cancel' te klikken.

- Wat is de return value van de confirm functie als de gebruiker op een van de buttons klikt?

Zet nu de return value van de prompt call op de console als je een tekst intypt en op 'ok' drukt.

- Wat is de return value van de prompt functie als de gebruiker een tekst intypt en op 'ok' klikt?
- Wat is de return value van de prompt functie als de gebruiker op 'cancel' klikt?

Elementen uit de DOM-tree opvragen

Een javascript programma dat in een browseromgeving uitgevoerd wordt, kan toegang krijgen tot de DOM-tree van de huidige pagina via een (voorgedefinieerde) '**document**' globale variabele.

De onderdelen van de DOM-tree noemt men nodes. Men gebruikt dezelfde terminologie die we reeds eerder zagen : parent nodes, child nodes, ancestor nodes, descendent nodes, etc.

Veel van die nodes komen overeen met HTML elementen, maar er zijn ook andere soorten nodes (bv. text nodes voor de teksten tussen begin- en eindtag van een element, of comment nodes voor stukjes commentaar uit het HTML document).

Een verwijzing naar een elementnode uit de DOM-tree kan je (onder andere) op de volgende drie manieren bekomen :

- **document.getElementById(**"abc"**);**

  doorzoekt de DOM-tree naar het element met id="abc" en retourneert een verwijzing naar dit element. Indien geen element wordt gevonden, retourneert de functie de null waarde.

- **document.getElementsByClassName(**"xyz"**);**

  doorzoekt de DOM-tree naar elementen met class="xyz" en retourneert een nodeList met verwijzingen naar deze elementen. Een nodeList is op dezelfde manier te gebruiken als een array, dus wat ons betreft is het verschil niet belangrijk. Indien geen enkel element wordt gevonden, retourneert de functie een lege list (net als een leeg array dus).

- **document.getElementsByTagName(**"img"**);**

  doorzoekt de DOM-tree naar img elementen en retourneert een nodeList met verwijzingen naar deze elementen. Een nodeList is op dezelfde manier te gebruiken als een array, dus wat ons betreft is het verschil niet belangrijk. Indien geen enkel element wordt gevonden, retourneert de functie een lege list (net als een leeg array dus).

De browser bouwt de DOM-tree op terwijl hij het HTML-document overloopt en analyseert. Als de browser daarbij een <script> elementen tegenkomt, zal hij dit meteen uitvoeren. Het kan dus gebeuren dat een stuk javascript wordt uitgevoerd terwijl de DOM-tree nog niet afgewerkt is! Indien deze code iets met de DOM-tree wil doen, kan dit dus problemen geven.

Code die de DOM-tree uitleest of aanpast, mag pas uitgevoerd worden wanneer de DOM-tree volledig is opgebouwd door de browser!

Om te bekomen dat de browser bepaalde code pas uitvoert eenmaal te DOM-tree klaar is, kun je onderstaand fragment gebruiken :

```
const setup = () => {
   ...
}
...
window.addEventListener("load", setup);
```

De code in de setup function zal pas worden uitgevoerd als de DOM-tree klaar is voor consumptie.

Opdracht 02

Probeer elk van de bovenstaande manieren uit waarmee je elementen uit de DOM-tree kunt opvragen. Doe dit op het console venster en gebruik hiervoor de pagina op

https://www.nieuwsblad.be/

Je zult natuurlijk zelf een passende id en className moeten zoeken in de DOM-tree van die pagina, deze uit de voorbeelden zullen hoogstwaarschijnlijk geen resultaten opleveren omdat die namen niet gebruikt worden (maar probeer deze ook eens uit, zodat je ook eens een lege nodelist ziet).

De innerHTML property

Je kan een element een andere inhoud geven door diens **.innerHTML** property een nieuwe waarde te geven.

Je zult dus eerst een verwijzing naar het element uit de DOM-tree moeten bemachtigen en dan diens .innerHTML instellen.

Opdracht 03

Maak een nieuw project met de volgende inhoud in het html bestand (in de body) :

```
<p id="txtOutput">Hello world!</p>
```

Stop de volgende code in de setup functie van je javascript bestand :

```
let pElement=document.getElementById("txtOutput");
pElement.innerHTML="Welkom!";
```

Laad de pagina in en ga na dat de tekst op het scherm niet "Hello World!" is, maar "Welkom!"

Opdracht 04

Zet een breakpoint op de eerste regel van de setup functie en herlaadt de pagina.

Vergewis je ervan dat er wel degelijk "Hello world!" op het scherm staat.

Kijk in de developer tools naar de DOM-tree en zoek de node van het <p> element, daarin zie je weerom "Hello world!".

Stap met de debugger doorheen de code totdat de regel uitgevoerd is waarin "Welkom!" wordt toegekend aan de innerHTML property van het <p> element.

Ga na dat er nu "Welkom!" op het scherm staat en dat dit ook in de DOM-tree veranderd is.

Opdracht 05

Verander het voorgaande programma zodanig dat de tekst pas gewijzigd wordt als je op een button met opschrift "Wijzig" drukt. Je kunt je hiervoor op het rekenmachine voorbeeld baseren.

Tekstvelden uitlezen

Je kan de tekst in een tekstveld bemachtigen via de 'value' property van het corresponderende element uit de DOM-tree.

Je zult weerom eerst een verwijzing naar het element uit de DOM-tree moeten bemachtigen en diens .value opvragen.

Demonstratie

In de .zip files 'demonstratie images.zip' en 'demonstratie lijsten.zip' vind je twee voorbeelden van het gebruik van de .innerHTML en .value properties.

In beide demonstraties wordt met javascript inhoud aan een pagina toegevoegd.

Opdracht 06

Maak een nieuw project met de volgende inhoud in het html bestand (in de body) :

```
<input id="txtInput" type="text" />
<input id="btnKopieer" type="button" value="Kopieer" />
```

Stop de volgende code in de setup functie van je javascript bestand :

```
let btnKopieer = document.getElementById("btnKopieer");
btnKopieer.addEventListener("click", kopieer);
```

Definieer ook een 'kopieer' functie in je javascript bestand :

```
const kopieer = () => {
    let txtInput = document.getElementById("txtInput");
    let tekst = txtInput.value;
    console.log(tekst);
}
```

Laad de pagina in, typ wat tekst in het tekstveld, klik op 'Kopieer' en ga na dat de ingetypte tekst op de console verschijnt.

Opdracht 'kopieer'

Voeg een <p> element toe aan de html code van het vorige programma :

```
<p id="txtOutput">geen output</p>
```

Breid het programma uit zodat de ingetypt tekst niet op de console getoond wordt maar in dit <p> element terechtkomt.

Opdracht 'substring'

Open 'opdracht substring screenshots.zip' en bekijk de twee afbeeldingen, deze tonen de toestand voor en na het klikken op de substring knop.

Bestudeer de documentatie van de substring function voor strings en schrijf een programma dat de werking van deze function demonstreert, zoals getoond in de screenshots.

Bv. als de gebruiker 'appelboom', 2 en 5 invult (zie screenshot VOOR) dan verschijnt er "pel" in de output paragraaf (zie screenshot NA).