Javascript – deel 02

In deze les komen nog enkele basiselementen van Javascript aan bod en bespreken we hoe je een project met Webstorm kan organiseren en bewerken.

Gebruikte software tools downloaden en installeren

Om onze HTML, CSS en Javascript broncode te bewerken gebruiken we Webstorm.

- Download en installeer Webstorm via https://www.jetbrains.com/webstorm/
- Vraag een licentie aan via https://www.jetbrains.com/student/
    - klik op 'apply now' en vul de form in met je @hogent.be email adres
    - je krijgt dan een email met verdere instructies
    - indien je geen licentie hebt kun je het programma slechts 30 dagen uitproberen!

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.

Video : Demonstratie Webstorm

Bekijk de opname 'video demonstratie webstorm – 29min.mp4', daarin wordt o.a. uitgelegd hoe je de files van een webpagina in Webstorm kunt organiseren.

Opdracht : Leeg project

Maak een nieuw leeg project (of nieuwe lege folder) aan in Webstorm, met daarin twee subfolders :

- subfolder 'styles', voor .css bestanden
- subfolder 'scripts', voor .js bestanden

Maak nu een HTML file index.html die verwijst naar een leeg index.css bestand en een leeg index.js bestand. Plaats deze lege bestanden telkens in de juiste subfolder.

In het vorige deel werd uitgelegd hoe je een Javascript bestand aan je webpagina kunt koppelen met het <script> element.

Stop in het index.js bestand de volgende code :

```
const setup = () => {
        // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
}

window.addEventListener("load", setup);
```

Dit leeg project kun je als startpunt gebruiken van de latere opdrachten die je zult maken.

Video : Demonstratie Javascript deel 02

Bekijk aandachtig de opname 'video demonstratie javascript deel 02 – 45min.mp4' en probeer de belangrijke stukken uit op je eigen installatie. Doe gewoon mee met de video en imiteer de handelingen.

De bijbehorende broncode vind je in volgende .zip bestanden :

- demo leeg project.zip
- demo rekenmasjien.zip

Belangrijk

Javascript heeft meerdere manieren om functies (methods) te definiëren, bv. deze drie definities introduceren allen een functie PrintHelloWorld zonder parameters :

De nieuwe manier (met arrow functies)	De oudere manieren (met *function* keyword)	

const PrintHelloWorld = () => {
  console.log("Hello world!");
}
	function PrintHelloWorld() {
	  console.log("Hello world!");
	}
		const PrintHelloWorld = function () {
		  console.log("Hello world!");
		}

de term 'arrow' slaat trouwens op het '=>' stukje

En ook voor variabelen zijn er meerdere mogelijkheden, bv. met de 'var' en 'let' keywords :

```
var tekst1 = "Hello World";      // de oudere manier
let tekst2  = "Hello World";     // de nieuwe manier
```

In het wild kun je alle vormen tegenkomen, maar in deze cursus spreken we af :

- We gebruiken altijd **'let'** om variabelen te declareren en geen 'var', dit sluit beter aan bij hoe declaraties werken in moderne programmeertalen zoals C#.

- We definieren onze functies (methods) altijd als **arrow functions** :

```
const setup = () => {
  ...
}
```

en niet meer op de oudere manieren :

```
function setup() {
  ...
}
```

of

```
const setup = function() {
  ...
}
```

- We zullen geen 'concise body arrow functions' gebruiken maar alles voluit schrijven (dus steeds met een stuk code tussen accolades).

Scope

Ga naar de "Introduction to Javascript" cursus op Codecademy

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

En doorloop de volgende sectie

4. Scope

Globale vs. lokale variabelen

Globale variabelen zijn variabelen die **niet** in een function worden gedeclareerd.

Lokale variabelen zijn variabelen die **wel** in een function worden gedeclareerd.

Bijvoorbeeld,

```
let globalVar="dit is een globale variabele";

const mijnFunctie = ()  => {
  let localVar="dit is een lokale variabele";
}
```

Er is een groot verschil tussen variabelen met 'var' dan wel met 'let' te introduceren. In het algemeen is het effect van 'let' intuïtiever, veiliger qua potentiele programmeerfouten en meer in lijn met hoe variabelen werken in talen als Java en C#.

In deze cursus gebruiken we altijd 'let' voor zowel lokale als globale variabelen.

Tijdens het debuggen in de Chrome Developer Tools vind je deze terug onder 'Scope' :

- lokale variabelen in de 'Local' sectie
- globale variabelen in de '**Script**' sectie (let op, niet de 'Global' sectie!)

Bij programma's die uit meerdere javascript files bestaan kunnen globale variabelen tot zeer subtiele (en moeilijk te debuggen) fouten leiden, probeer dit dus te vermijden.

Introduceer in je programma's geen vermijdbare globale variabelen!

Alle globale variabelen in een programma (ongeacht uit welke .js file) komen allen op eenzelfde hoop terecht, waardoor naamsconflicten mogelijk worden. Verschillende stukken code (bv. van jezelf en van een library die je gebruikt) zouden dus elkaars globale variabelen kunnen overschrijven als ze per ongeluk dezelfde naam gebruiken.

Arrays

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

Ga naar de "Introduction to Javascript" cursus op Codecademy

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

En doorloop de volgende sectie

5. Arrays

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 List uit C# or ArrayList uit Java. Er zijn methods voorhanden om makkelijk inlassingen, toevoegingen of verwijderingen te realiseren en hun grootte (het aantal slots) kan varieren.

Bovendien kan een javascript array ook verschillende soorten waarden bevatten, waarbij bv. in slot 7 een string voorkomt terwijl slot 11 een getal bevat. In een programmeertaal als C# kan dit niet : ofwel bevatten de slots in een array strings ofwel getallen, maar nooit een mengeling van beide. Deze flexibiliteit van javascript arrays lijkt handig, maar levert in de praktijk niet veel voordeel op.

Javascript gebruikt net als veel andere programmeertalen de rechte haakjes [ ] voor arrays. Dit maakt het natuurlijk verwarrend omdat ze er bv. uitzien als een C# array, maar zich gedragen als een C# List.

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"
```

Merk op dat je in javascript geen error krijgt als je een ongeldige index gebruikt, bv. index -1 of 3 in het voorbeeld hierboven (omdat het strikt genomen geen ongeldige index is, denk eraan dat arrays geen vaste grootte hebben in javascript!).

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)

splice(...)

een ietwat schizofrene functie waarmee je elementen in een array kunt inlassen of verwijderen
zie http://www.tutorialspoint.com/javascript/array_splice.htm

Loops

Ga naar de "Introduction to Javascript" cursus op Codecademy

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

En doorloop de volgende sectie

6. Loops

Hiermee hebben we de belangrijkste Javascript taalconstructies gezien die we dit jaar zullen nodig hebben.

Opdracht vingeroefening

Voor onderstaande oefeningen zul je in Webstorm een quasi lege HTML pagina moeten maken die naar een javascript file met code verwijst. Noem de files resp. index.html en index.js.

> *Als Webstorm fouten geeft bij het gebruik van 'let', moet je de Javascript versie verhogen. Ga naar File->Settings ->Languages&Frameworks ->Javascript en kies "ECMAScript 6" als versie.*

Open de index.html file in je browser en bekijk de output op de console. In Chrome kun je het console venster tevoorschijn halen via de Developer tools (tabblad Console).

Let op : telkens je iets aanpast in je Javascript code in Webstorm, zul je in je browser de pagina moeten herladen zodat deze de nieuwste versie van je code gebruikt.

Let op : Het risico bestaat dat je browser bij het herladen een gecachete versie van je Javascript file gebruikt (waardoor je niks merkt van je laatste code wijzigingen). Bij twijfel, kies voor 'Hard Reload'.

a) Schrijf een simpel programma dat "Hello world" op de console zet. Pas daarna in Webstorm de tekst aan naar "Hello world!!!!!!!!!!" en probeer opnieuw.

b) Schrijf een simpel programma dat een vierkant van sterretjes produceert op de console, bv.

```
****
****
****
****
```

(tip : gebruik een geneste lus)

Het aantal regels is instelbaar via een globale constante 'aantalRegels'.

c) Schrijf een simpel programma dat alle 3-vouden en 5-vouden onder 500 op de console zet en op het einde ook hun som toont.

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.

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 niet enkel output bekijken, maar ook opdrachtjes intypen en uitvoeren, handig om snel iets uit te proberen!

Type bv. eens onderstaande regels code in het console venster van je browser :

```
let naam = "Jan";
console.log("De naam is "+naam);
```

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. M.a.w. de output van je pagina zal in de HTML-pagina moeten terechtkomen zodat de gebruiker dit kan zien.

Opdracht input/output

Probeer eerst eens elk van de drie soorten popups uit op de console in je browser.

Zet daarna de return value van de confirm call op de console en probeer uit door eens op 'ok' te klikken. Probeer vervolgens ook eens uit door 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?