Javascript – deel 01

Deze les introduceert javascript en toont hoe je programmatorisch de DOM-tree kunt manipuleren.

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 waarin je

- voor elke uitprobeer opdracht een entry maakt met screenshots ter staving van wat je deed
- je antwoorden op de gestelde vragen neerschrijft

Oplossingen van grotere opdrachten (met veel code) bewaar je apart in een .zip file.

Javascript

Javascript is een programmeertaal en bijbehorende run-time omgeving, net als bv. Java en de Java virtuele machine of C# en de .Net run-time.

Merk trouwens op dat Javascript niks te maken heeft met Java, op de gelijkaardige naam en het gebruik van accolades voor code blocks na. De code ziet er oppervlakkig misschien vertrouwd uit, maar het is echt een heel andere manier van programmeren dan met Java of C#.

Javascript kan als stand-alone omgeving gebruikt worden, zoals bv. Node.js bewijst, maar wij zullen ons concentreren op het gebruik ervan om gedrag aan webpagina's toe te voegen. Onze derde pijler (naast structuur en opmaak)

Dit gedrag verkrijgen we door stukjes code aan gebeurtenissen te koppelen (bv. een klik op een element) zodat die code wordt uitgevoerd op het gewenste moment.

Zulke code kan bv. wijzigingen in de DOM-tree aanbrengen zoals

- elementen toevoegen of verwijderen
- de CSS-classes of CSS-properties van een element wijzigen
- http requests versturen op de achtergrond en de response verwerken (zie AJAX)

Elke browser bevat dus een Javascript run-time component die javascript code kan uitvoeren en geeft toegang tot de DOM-tree van de huidige pagina. Een javascript programma wordt door de browser ingeladen en uitgevoerd omdat ernaar verwezen wordt vanuit de actuele HTML pagina.

Een javascript programma behoort altijd tot de context van de pagina waarin het uitgevoerd wordt. Wanneer de gebruiker naar een andere pagina navigeert, stopt de uitvoering van het actuele programma. Eenmaal de nieuwe pagina is ingeladen wordt dan de javascript code van die nieuwe pagina gestart (zelfs indien het dezelfde javascript code als op de vorige pagina was).

Het <script> element

Om een stuk javascript code aan een HTML document te koppelen, moet je in het HTML document een <script> element toevoegen. Dit element kan ofwel

- de eigenlijke code bevatten tussen begin- en eindtag
- naar code verwijzen in een extern javascript bestand (extensie .js doorgaans)

Net als bij onze CSS regels, opteren we ervoor om de javascript code in een apart bestand te plaatsen.

Wanneer de browser het <script> element tegenkomt bij het verwerken van het HTML document, wordt de javascript code meteen geladen en uitgevoerd. Terwijl dit gebeurt wacht de browser met het inladen (en renderen) van andere resources zoals bv. afbeeldingen, wat niet zo prettig is voor de gebruikers.

Oplossing : stel de uitvoering zo lang mogelijk uit door <script> pas net voor de </body> tag te plaatsen.

Een HTML document kan trouwens meerdere <script> elementen bevatten, deze worden dan na elkaar uitgevoerd.

Opdracht 01

Als eerste kennismaking zullen we gebruik maken van de Codecademy cursus over Javascript.

https://www.codecademy.com

Dit is een online leerplatform waarmee je a.d.h.v. opdrachten telkens iets bijleert over een gekozen onderwerp. Hun aanbod bevat trouwens veel meer onderwerpen dan enkel Javascript.

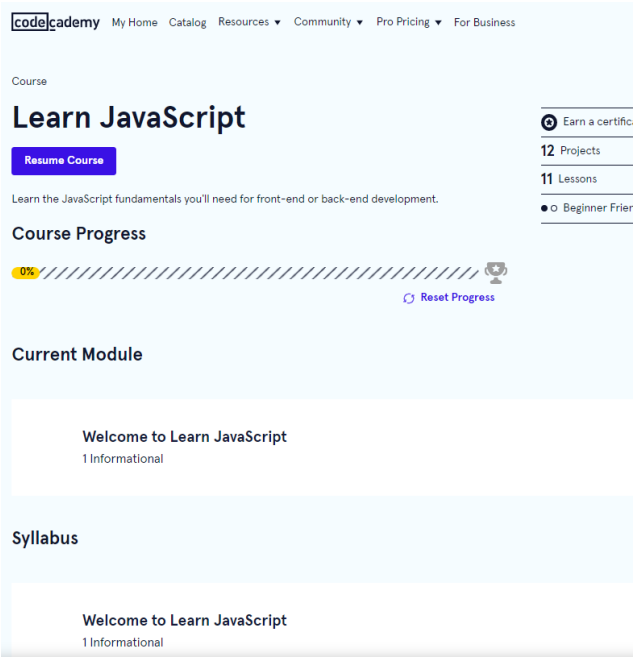
Maak een account aan op deze site en ga naar de "Introduction to Javascript" cursus

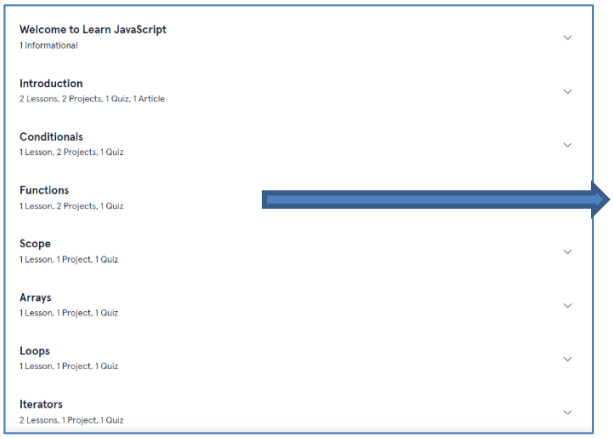
https://www.codecademy.com/learn/introduction-to-javascript

Doorloop de volgende secties in de cursus en maak de opdrachten die je voorgeschoteld krijgt.

De "PRO" secties zijn betalend en kun je overslaan. Je moet enkel de lessons Lesson volgen

Het is belangrijk dat je de opdrachten doorloopt terwijl je ingelogd bent, alleen dan wordt je vordering doorheen de cursus bijgehouden en kun je dit desgevraagd aan je docent voorleggen.





Van stap 1 t.e.m. stap 8 wordt de klassieke manier uitgewerkt van functies. Wij zullen de Arrow notatie gebruiken volgens ES6

In deze oefenreeks doorloop je de volgende onderdelen:
“Introductio”, “Conditionals”, “Functions” en “Scope”

Merk op dat we in deze cursus een functie als volgt declareren :

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

en niet meer op de oude wijze :

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

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 functie1 = () => {
  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' intuitiever, 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 zo weinig mogelijk globale variabelen!

Alle globale variabelen in een programma (ongeacht uit welke .js file) komen allen op eenzelfde hoop terecht, waardoor naamconflicten 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.

Opdracht : Leeg project

Bekijk eerst het document: "File and Code templates.pdf"

Maak een nieuw leeg project aan in Webstorm met daarin twee subfolders

- folder 'styles', voor .css bestanden
- folder '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.

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 opdrachten die je zult maken.

Javascript deel 01

Bekijk ook de bijbehorende .zip bestand

- -demo rekenmachine.zip