Javascript – deel 01

Deze les introduceert Javascript en doorloopt enkele secties uit een Codecademy cursus die de basiselementen van de taal te demonstreren : variabelen, if/else structuren, functies (methods).

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 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 omgeving.

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 in C# of Java.

De Javascript run-time omgeving is ofwel

- losstaand
    - Bijvoorbeeld Node.js kan Javascript programma's uitvoeren en wordt gebruikt voor console applicaties en webserver software.
- ingebouwd in je browser
    - Een HTML pagina verwijst doorgaans naar Javascript broncode. De browser zal deze code uitvoeren zodra de pagina is ingeladen. De code krijgt van de browser toegang tot de inhoud van de pagina. We kunnen de inhoud van het tabblad en de pagina aanpassen, door ingebouwde functies (methods) op te roepen.

Wij richten ons in deze cursus op de browser omgeving en haar mogelijkheden om de pagina te manipuleren. Bijvoorbeeld hoe we inhoud kunnen toevoegen (of aanpassen), wat we zoal kunnen doen met de styling van elementen en hoe we kunnen inspelen gebruikersacties (scrollen, klikken, ...).

In het kort : ons programma koppelt stukjes code aan pagina-gebeurtenissen. Telkens die gebeurtenis zich voordoet in de pagina (bv. een klik op een <img> element) zal ons gekoppeld stukje code worden uitgevoerd.

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 in de pagina (zie AJAX)

Elke browser bevat dus een Javascript run-time component die Javascript code kan uitvoeren en ons zo toegang geeft tot de DOM-tree van de huidige pagina. Een Javascript programma wordt door de browser ingeladen en uitgevoerd omdat ernaar verwezen wordt vanuit het HTML document. Concreet, om een Javascript programma 'program.js' in pagina index.html te laten uitvoeren, moet index.html een <script> verwijzing naar 'program.js' bevatten.

Een uitvoerend 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.

Dat lijkt handig **maar** terwijl onze Javascript code uitvoert, wacht de browser met het inladen (en tonen) van andere resources zoals bv. afbeeldingen, wat niet zo prettig is voor de gebruikers.

Oplossing : we stellen de uitvoering zo lang mogelijk uit door het <script> element pas net voor de </body> tag te plaatsen. Quasi helemaal op het einde van het document dus, waardoor het als een van de laatste elementen zal worden verwerkt.

Een HTML document kan trouwens meerdere <script> elementen bevatten, deze worden dan netjes na elkaar uitgevoerd zodra de browser ze tegenkomt in het HTML document.

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. Let erop dat je NIET ingaat op hun 7-dagen gratis “PRO” aanbod, anders kan het zijn dat je in de problemen raakt vanaf dag 8 (sommige content is dan niet meer bereikbaar, terwijl die wel beschikbaar blijft als je niet ingaat op hun gratis proefaanbod).

Ga naar de "Introduction to Javascript" cursus

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

Doorloop de eerste drie secties in de cursus

1. Introduction
2. Conditionals
3. Functions

en maak de opdrachten die je voorgeschoteld krijgt. De "PRO" secties zijn betalend en sla je best over.

Tips :

- Het is belangrijk dat je de opdrachten doorloopt terwijl je ingelogd bent, alleen dan wordt je vordering doorheen de Codecademy cursus bijgehouden en kun je dit desgevraagd aan je docent voorleggen.
- Telkens je een opdracht gemaakt hebt moet je op ‘Run’ klikken om je oplossing uit te proberen. De website laat je pas verdergaan met ‘Next’ als je oplossing correct is.
- Als je niet verder raakt maar je oplossing lijkt correct : ga eens na of je exact deed wat ze verlangden want "iets gelijkaardigs doen" wordt vaak niet aanvaard. Bijvoorbeeld bij strings moeten de hoofd- en kleine letters ook precies zo worden overgenomen.
- Als je echt vast zit : de hints staan in de linker zijbalk, onder de opdrachten.

In de volgende les zullen we nog een paar secties uit de Codecademy cursus doornemen.

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).