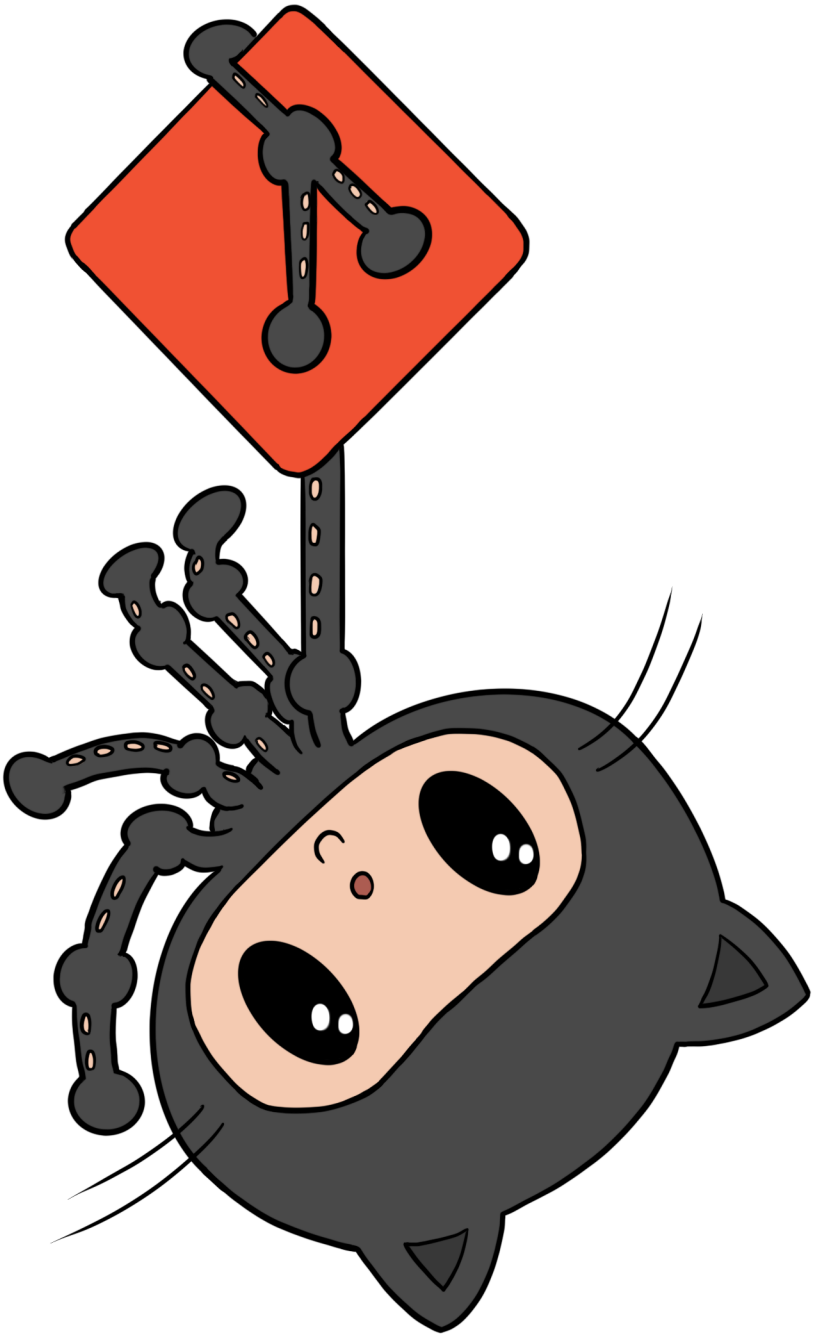
Javascript

Tim Quax (tquax@novacollege.nl)



Contents

JavaScript (JS)

JavaScript is een veel-gebruikte taal om o.a. websites (die je maakt met HTML en CSS) *dynamisch* te maken. Dynamisch betekent dat de website meer kan doen dan alleen maar op je browser-scherm stilstaan (dat is statisch).

Denk bijvoorbeeld aan een refresh-knop, of een foutmelding als je e-mail adres een typfout bevat, of een melding dat je moet opschieten met bestellen want de voorraad is bijna op.

In deze reader ga je leren over JavaScript, maar ook JSON: JSON is een manier om informatie neer te zetten; een manier dat mensen EN programmeertalen goed kunnen snappen. Dat wordt vaak gebruikt als doorgeefluik. Bijvoorbeeld:

- JavaScript doet een verzoek aan PHP;
- PHP haalt vervolgens informatie uit de database en geeft dat terug in JSON;
- JavaScript ontvangt dat en kan precies zien welke informatie terug is gegeven;
- Javascript plukt de informatie eruit die het nodig heeft en zet dat neer in de HTML.

Benodigdheden

- Editor (PHPStorm bijvoorbeeld)
- PHP (XAMPP bijvoorbeeld).
- Browser (Firefox of Chrome bijvoorbeeld).
- Google (Begrijp je iets niet? Dit is je beste vriend!)

Front-end en Back-end

Dit zijn termen die vaker voor gaan komen, gezien je werkt met HTML/CSS/JavaScript zowel als PHP: - Front-end is de kant die je ziet: HTML, CSS maar ook Javascript - Back-end is de kant die je niet ziet: PHP, Java, C# wat draait op een laptop of server.

Geschiedenis van JavaScript

JavaScript is gemaakt in 10 dagen, in 1995. Laat dat even inzinken. Tien dagen. Door 1 persoon, die destijds werkte voor het bedrijf Netscape (de browser).

JavaScript is initieel ontworpen als script taal in de browser, om zo soepel mogelijk samen te kunnen werken met Java. De naam JavaScript komt daar ook vandaan. Waarom Java specifiek, als het ook werkt met alle andere talen? Heel simpel: Java was destijds een van de populairste programmeertalen, en kon zelfs draaien in browsers als mini-applicaties (*applets*). Op die populariteit kan je dan mooi meeliften.

Tegenwoordig zijn er enorm veel bedrijven en organisaties die meewerken aan het ontwikkelen van JavaScript. De taal veranderd enorm snel en is heel erg flexibel. De laatste jaren is JavaScript zelfs ook populair als *back-end*. Je kan het op dezelfde manier draaien als b.v. PHP, Java en C# met behulp van de V8 engine, gemaakt door Google.

Hoofdstuk 1: Kennis maken met JavaScript

Omdat we JavaScript gaan gebruiken om een webpagina aan te passen, moeten we eerst een webpagina hebben. We gaan een stuk HTML en CSS schrijven en dat vervolgens veranderen met JavaScript.

1.1 HTML & CSS

We beginnen door onderstaande pagina te maken:

Maak een nieuw project in `PHPStorm` en maak het bestand `index.html` aan met deze inhoud:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JavaScript hoofdstuk 1</title>
    <link rel="stylesheet" href="website.css">
  </head>
  <body>
    <div id="box1" class="box"></div>
    <div id="box2" class="box"></div>
    <script src="website.js"></script>
  </body>
</html>
```

Er staan twee dozen in de HTML: *box1* en *box2*. Die gaan we mooier maken. Maak `website.css` aan met:

```
#box1 {
    background-color: red;
    margin-left: 300px;
}
#box2 {
    background-color: blue;
    margin-left: 200px;
}
.box {
    height: 200px;
    width: 200px;
    margin-top: 300px;
    float: left;
}
```

Als je alles goed hebt overgenomen ziet deze pagina er nu exact uit als het voorbeeld hierboven.

1.2 JavaScript

Om ons eerste stukje JavaScript te schrijven maken we gebruik van het JavaScript bestand die in de HTML staat vermeld:

```
<script src="website.js"></script>
```

Dat zegt eigenlijk: Laad een script in, genaamd `website.js`. Die gaan we nu aanmaken met de inhoud:

```
document.querySelector('#box1').style.backgroundColor = "green";
```

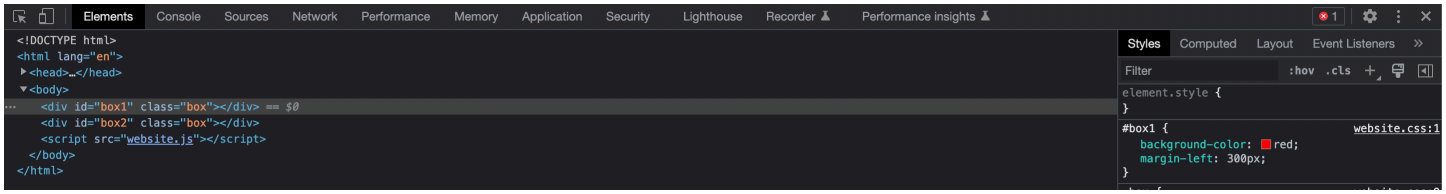
Laten we dat even ontleden. JavaScript is een taal waarin je een heleboel aan elkaar kan plakken. Als een functie iets teruggeeft waar functies in zitten, kan je die meteen uitvoeren. Dus:

- `document`, daar staat de hele website in, inclusief css, en functies om ermee te werken.
- `.querySelector('#box1')` is een functie in `document` om HTML elementen te selecteren. Wat we dan gaan pakken (of *queryen*) is een `selector` - exact hetzelfde als je in CSS gebruikt. In dit geval het ID `box1`.
- `.style` pakt alle stijlen op het element die we pakken (of *queryen*).
- `.backgroundColor` is specifiek de stijl voor de achtergrondkleur. Die kan je bekijken of aanpassen.
- `= "green";` maakt de achtergrondkleur groen (net zoals in CSS `#box1 { background-color: "green"; }` dat zou doen). Hier kan je ook *hexadecimale kleuren* zetten, zoals `#00FF00`.

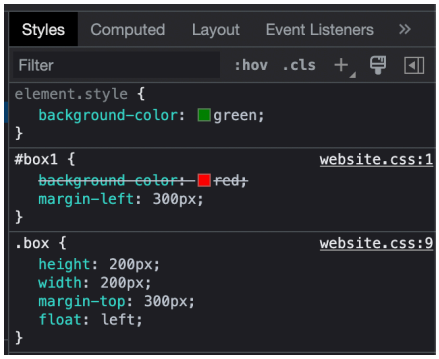
Als je nu de website bekijkt, zie je dat doos nummer 1 groen is geworden. Het werkt!

1.3 Developer tools

Fijn dat het nu goed gaat, maar wat moeten we doen als er iets mis gaat? Hoe controleer je dat? *Developer Tools* is je redding. Druk op F12 en er verschijnt een balk onderin:

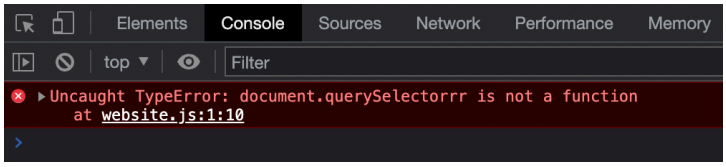


Alle HTML van de hele pagina staat aan de linkerkant. Als je iets aanklikt zie je de styling staan in het vak helemaal rechts. Daar kan je ook zien welke stijlen worden overgeschreven. In ons geval door JavaScript, en dat ziet er zo uit:



Dat kan je lezen als: op het element zelf is een stijl neergezet (door JavaScript). Dat overschrijft de `background-color` stijl die het element krijgt vanaf `#box1` (daar staat ook een streep door). Helemaal rechts staat waar de stijl vandaan komt, in dit geval `website.css` op regel 1.

Alle foutmeldingen van JavaScript komen in de tab `Console`. Als je `console.log("test")` in JavaScript zet ergens, komt dat ook hier terecht (dat is de echo / Console.WriteLine van JavaScript). Als we bijvoorbeeld een typfout hebben gemaakt in onze code, dan zie je dit staan:



1.4 Eindopdracht hoofdstuk 1

Box1 is inmiddels aangepast, probeer nu zelf de kleur van box2 aan te passen naar *oranje* met een *hexadecimale kleurcode*. Gelukt? Voeg dan nog twee divs toe in de HTML en pas daar de kleur van aan met JavaScript.

Hoofdstuk 2: Nog meer JavaScript

Laten we nog wat meer aspecten van JavaScript doorlopen

2.1 Event listener: on load

Het kan voorkomen dat je JavaScript iets wilt laten doen zodra de pagina voor het eerst verschijnt. Of als je ergens op klikt. Of als je op enter drukt. Al deze voorbeelden (en meer) noem je *events*. Op elk HTML element (en zelfs de hele pagina) kan je aandachtig luisteren tot een specifiek *event* aan de orde is.

Bijvoorbeeld om te kijken of de hele pagina geladen is:

```
window.addEventListener("load", function () {
    console.log("De pagina is geladen!");
});
```

Als het JavaScript bestand bijvoorbeeld wordt ingeladen in de `<head>` van je HTML document, dan draait het voordat alle HTML werkelijk in de pagina is neergezet. Dan zou je dus een foutmelding krijgen bij het wijzigen van de achtergrondkleur van `#box1`, want die bestaat nog niet. De `console.log` hierboven verschijnt pas als de hele pagina compleet geladen is.

Laten we het ontleden:

- `window` is het browser scherm
- `addEventListener(` is de functie die dan gaat luisteren op `window` voor een specifiek *event*.
- `"load"` is de naam van het *event*. Dus zodra het geladen is.
- `function () {` is een *naamloze functie*, die wordt uitgevoerd door `addEventListener` zodra het klaar is met luisteren. Daarom heb je ook geen naam nodig - de hele functie geef je mee aan de event listener. Een korte versie van een naamloze functie is `() => {`.
- `console.log()` toont een bericht in de *Developer Tools console*.

2.2 Event listener: on click

Een ander voorbeeld is iets uitvoeren als je klikt op de `<div>` met ID `box1` :

```
document.querySelector('#box1').addEventListener('click', function() {
    console.log("Ooooh je mag niet klikken!");
});
```

Laten we het toch weer even ontleden:

- `document` , bevat de hele website, html en css.
- `.querySelector('#box1')` selecteert een HTML element adhv een `selector` - dezelfde manier als CSS. `#box1` is dus het HTML element met ID `box1` .
- `addEventListener(` is de functie die dan gaat luisteren op `#box1` voor een specifiek *event*.
- `"click"` is de naam van het *event*. Dus het klikken op het element.
- `function () {` is een *naamloze functie* die dan wordt uitgevoerd door `addEventListener` .
- `console.log()` toont een bericht in de *Developer Tools console*.

2.2 Event listener: on submit

Je kan ook een handeling uitvoeren als je een formulier invult en verstuurd. Dat doe je door op een submit knop te drukken of op enter drukken in een tekst vak, oftewel:

```
document.querySelector('#search_form').addEventListener('submit', function() {
    console.log("Laten we dan nu het hele formulier naar PHP sturen.");
});
```

2.3 Tekst of HTML toevoegen

Je kan meer doen dan alleen de stijl van een HTML element aanpassen. Laten we eens wat tekst, en daarna HTML, toevoegen. Aan `box2` dit keer, `box1` heeft genoeg actie gezien.

Een stuk tekst in de `<div>` met ID `box2` zetten doe je zo:

```
document.querySelector('#box2').innerText = "Dit is nu de tekst in Box Twee.";
```

Het lijkt heel erg op de stijl-verandering. In plaats van de `style` veranderen we nu de `innerText` variabel, en dat wordt meteen in de HTML doorgevoerd.

Een stuk HTML in `#box2` zetten doe je zo:

```
document.querySelector('#box2').innerHTML = "<h1>Dit is nu</h1><p>de tekst in Box Twee.</p>";
```

Bijna geheel hetzelfde als de tekst veranderen. Het grote verschil is dat je absoluut geen HTML kan toevoegen met `innerText` , dat wordt dan exact getoond zoals je het typt.

2.4 Eindopdracht hoofdstuk 2

Maak `website.js` zodanig dat:

- `box2` krijgt een stuk tekst met html met een mooie achtergrond kleur als je klikt op `box1` ;
- `box1` krijgt hetzelfde maar dan met een mooiere achtergrond kleur, als je klikt op `box2` .
- Beide deze brokken code draaien pas zodra het hele scherm geladen is.