Coderdojo javascript projecten

Sam Clercky

4 september 2022

Hoofdstuk 1

Cheatsheet

1.1 HTML

1.1.1 Structuur

Elk **HTML** bestand begint met volgende tekst. Dit vertelt de computer dat we **HTML** willen gebruiken en geeft extra informatie over hoe de computer onze website moet weergeven.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Mijn website</title>
    </head>
    <body>
        Hier komt de rest van de html
    </body>
</html>
```

HTML bestaat uit tags. Een tag komt in 2 vormen voor:

- Met een begin en een eind die inhoud bevat:

  ```
  <div>inhoud van de tag</div>
  ```

- Zonder inhoud:

  ```
  <input />
  ```

Elke tag kan ook extra informatie bevatten via attributen. Deze zijn vaak optioneel maar handig.

- Met een begin en een eind die inhoud bevat:

```
<div id="mijn_id" class="mijn_klasse">
    inhoud van de tag
</div>
```

- Zonder inhoud:

```
<input id="mijn_id" type="password" />
```

Opmerking: HTML houd geen rekening met spaties, tabs, ... Dus volgende 2 notaties zijn hetzelfde:

```
<div
    class="blabla"
    id="blabla">
        Dit is indoud
</div>
<div class="blabla" id="blabla">Dit is indoud</div>
<div class="blabla" id="blabla">Dit is indoud met veel spaties
                        </div>
```

1.1.2 Voorbeeld rood vierkant

Plaatsen van een rood vierkant op een gekozen locatie:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Rood vierkant</title>
    </head>
    <body>
        <div style="
            background: red;
            width:100px;
            height:100px;
            position:absolute;
            left:100px;
            top:100px;
```

```
        "></div>
    </body>
</html>
```

Het grote verschil met de begincode is de **div**. En deze heeft 1 groot attribuut **style**. De waarde hierin is eigenlijk **CSS** waarover later meer.

Kort overlopen van de verschillende waarden in de **style**:

- *background*: achtergrond rood (red) maken
- *width*: breedte op *100px* zetten
- *height*: hoogte op *100px* zetten
- *position*: nodig om een niet standaard positie te kunnen opgeven
- *left*: afstand van de linker rand
- *right*: afstand van de top

Figuur 1.1: Resultaat rood vierkant

1.1.3 Belangrijke HTML tags

- *html*: Bevat alle andere **HTML** tags
- *head*: Normaal nooit zichtbaar, maar bevat belangrijke informatie over de **HTML**
- *body*: Alles in deze tag is zichtbaar op het scherm
- *title*: De titel van je website en komt enkel voor in een *head*-tag
- *div*: Een block-container zonder extra betekenis
- *img*: Voeg een afbeelding toe met de *src*-attribuut (url)
- *input*: Tekstveld, handig voor vragen van een spelersnaam

- *style*: **CSS** voor gans de webpagina
- *script*: **JavaScript** voor de webpagina

1.1.4 Belangrijke attributen

- *style*: Voeg **CSS** toe direct in de **HTML**
- *id*: Speciale naam voor de tag (moet uniek zijn voor elke tag)
- *class*: Zelfde als *id* maar moet niet uniek zijn
- *src* (enkel *img*): De url van de tonen afbeelding
- *value* (enkel *input*): De ingevulde waarde in het tekstveld
- *type* (enkel *input*): Soort input. Belangrijke mogelijke waarden:
  - *text* (Standaard waarde): normale tekst
  - *password*: wachtwoord
  - *number*: getal

1.2 CSS

1.2.1 Structuur

```
selector {
    eigenschap: waarde;
}
```

Er zijn 2 manieren om **CSS** toe te voegen. Ofwel via een *style*-tag in de *head*-tag:

```
...
<head>
    ...
    <style>
        selector1 {
            eigenschap: waarde;
        }
        selector2 {
            eigenschap: waarde;
        }
        selector3 {
```

```
            eigenschap: waarde;
        }
    </style>
</head>
...
```

Een andere manier is zoals eerder met de *style*-attributen. Het grote verschil is dat er nu geen nood meer is aan de selector:

```
...
<div style="
    eigenschap1: waarde;
    eigenschap2: waarde;
    eigenschap3: waarde;
" ></div>
...
```

Een selector is een manier om een aantal tags te selecteren en daarvan de eigenschappen te wijzigen. Er zijn 3 grote manieren van selecteren:

- Via tagnaam: bv: **div**: selecteert alle divs.
- Via classmaam: bv: **.klasse**: selecteert alle tags die `class="klasse"` hebben
- Via id: bv: `#id_naam`: selecteert de tag die `id="id_naam"` heeft. Dit is normaal altijd 1 tag.

1.2.2 Interessante eigenschappen

Hier is een lijstje van de meest interessante eigenschappen voor het opmaken met CSS. Alle eigenschappen veranderen iets van alleen de geselecteerde elementen!

- **background**: bepaalt de achtergrond
- **color**: bepaalt de tekstkleur
- **width**: bepaalt de breedte
- **height**: bepaalt de hoogte
- **position**: bepaalt de manier waarop de positie van het element wordt bepaalt. De belangrijkste zijn:
  - **absolute**: positie ten opzichte van de linker bovenhoek

  - **relative**: positie ten opzichte van de linker bovenhoek van de ouder (parent)
  - **fixed**: zoals absolute maar wanneer je scrolt, blijft het element op zijn plaats
- **left**: afstand van links
- **top**: afstand van de bovenkant
- **right**: afstand van rechts
- **bottom**: afstand van de onderkant
- **margin-{left,top,right,bottom}**: minimale afstand van links,boven,rechts,onder
- **padding-{left,top,right,bottom}**: minimale interne afstand van links,boven,rechts,o
- **opacity**: doorzichtbaarheid

De waarden die je in bovenstaande eigenschappen kunt invullen, worden hieronder opgelijst:

- **Afstand**: px (pixels), % (procent van de ouder)
- **Kleur**:
  - **Kleurnaam**: red (rood), green (groen), blue (blauw), black (zwart), white (wit)
  - **rgb(0-255, 0-255, 0-255)**: Kleur samenstellen uit r(ood), g(roen) en b(lauw)
  - **hsl(hue, saturation, lightness)**: Kleur samenstellen uit hue (tint), saturation (verzadiging), lightness (lichtheid)
  - **#XXXXXX**: Hetzelfde als rgb maar korter. Elke kleur bestaat uit 2 tekens tussen 0-9+A-F. Enkele voorbeelden:
    * Wit: #FFFFFF
    * Zwart: #000000
    * Rood: #FF0000
    * Groen: #00FF00
    * Blauw: #0000FF

1.3 JavaScript

1.3.1 Structuur

Om JavaScript de kunnen gebruiken, moet je deze in de HTML in een **script**-tag zetten

```
<script type="text/javascript">
    // Dit is een comment
    // Ik wordt nooit uitgevoerd, maar ben hier
    // om jou te helpen
    // In deze tag komt verder alle JavaScript
    console.log("Kijk ik ben cool, druk F12 om mij te zien!");
</script>
```

De meest algemene structuur die te vinden is in JavaScript is het **statement**

```
console.log("Een beetje veel tekst :)");
```

Een aantal dingen om op te merken:

- Elke statement bevat een actie (iets doen)
- Elke statement eindigt met een punt komma **;**

Je kunt een statement zien als 1 blokje in Scratch en veel statement na elkaar is alsof je veel blokjes aan elkaar kunt vastmaken.

Variabelen

In Scratch kun je ook variabelen maken. Dit waren stukjes geheugen waar je een waarde aan kon meegeven en zo een score kon bijhouden. In JavaScript doe je dit zo:

```
let naamVariabele = 1;
naamVariabele = naamVariabele + 1;
```

Een variabele aanmaken begint altijd met het woordje **let**. Hiermee zeg je tegen je computer dat hij een stukje geheugen moet reserveren en dat de naam **naamVariabele** moet geven.

Opmerking: Je kunt maar 1 keer een variabele aanvragen onder dezelfde naam in dezelfde scope (hierover later meer)! Anders gaat de computer klagen. Het is alsof je in een restaurant 2 reserveringen maakt onder dezelfde naam. Als je dan toekomt en ze vragen naar je naam, weten zij niet meer welke reservatie precies voor wie was.

Nu je een variabele hebt, kun je iedere keer je de waarde die je erin hebt opgeslagen, nodig hebt, gewoon de naam van de variabele gebruiken.

Met het =-symbool zeg je wat er in de variabele moet worden opgeslagen. Je kunt dus ook de originele waarde terug veranderen. Dit is wat er gebeurt op de 2de lijn. Eerst wordt **mijnVariabele** opgeteld bij 1 en wordt het resultaat opgeslagen in **mijnVariabele** (mijnVariabele is nu 2).

Soorten waarden dat je kunt opslaan in een variabele:

- `number` (nummer/getal): Vb 1,2,3, 1.234, -4, ...
- `string` (tekenreeks of tekst): `"Dit is een string"`
- `bool` (waar of niet waar): Kan slechts 2 waarden bevatten: `true, false`.
- `array` (lijst): Lijst van waarden: Vb: lijst van nummers: `[1,2,3,4]`.
- `object` (object): Alles wat je niet kunt beschrijver als een nummer of stuk tekst (Vb `document,window,console`)
- `null,undefined` (niets, ongedefinieerd): opslaan van dingen die je niet kunt opslaan of nog niet weet.

Conditioneel code uitvoeren

Algemene structuur:

```
if (testbool) {
    // uit to voeren als true
} else {
    // code uit te voeren als false
}
```

Als 1 kleiner is dat 2, print dan naar de console:

```
if (1 < 2) {
    console.log("1 is kleiner dan 2");
} else {
    console.log("1 is niet kleiner dan 2");
}
```

Je hoeft else niet altijd toe te voegen (optioneel):

```
if (1 < 2) {
    console.log("1 is kleiner dan 2");
```

```
}
console.log("Ik wordt altijd uitgevoerd");
```

Je kunt ook met variabelen werken:

```
let i = 1;
if (i < 2) {
    console.log("i is kleiner dan 2");
}
i = 3;
if (i < 2) {
    // Wordt niet uitgevoerd
    console.log("i is kleiner dan 2");
} else {
    console.log("i is niet kleiner dan 2");
}
```

Code herhalen

Structuur for i:

```
for (let i = 0; i < 10; i++) {
    console.log(i);
}
// Uitvoer:
// 0
// 1
// 2
// 3
// 4
// 5
// 6
// 7
// 8
// 9
```

Structuur for of

```
let mijnLijst = [1,2,3,4];
for (let nummer of mijnLijst) {
    console.log(nummer);
}
// Uitvoer:
```

```
// 1
// 2
// 3
// 4
```

Structuur while: zolang de conditie waar is, blijf herhalen

```
let i = 10;
while (i > 5) {
    console.log(i);
    i = i - 1;
}
// Uitvoer:
// 10
// 9
// 8
// 7
// 6
```

Groeperen van code

Met de vorige stukken is het nu al mogelijk om al heel complexe programma's te schrijven. Dit is ook hoe de eerste computers werden geprogrammeerd. Dit leidde echter tot het probleem dat code niet hergebruikt kon worden of het moeilijk werd voor een mens om het programma logisch nog te begrijpen.

Als antwoord hierop, werden functies (`function`) uitgevonden. Dit is een stuk code dat je een naam geeft, optioneel een aantal parameters en optioneel ook een variabele kan teruggeven. Deze functie kun je dan overal hergebruiken als je de naam kent.

Structuur van een functie:

```
function naamFunctie(parameter1, parameter2) {
    // Doe iets cool met parameter1 en parameter2
    let ietsCool = 1 + 1;
    return ietsCool;
}

// gebruiken van je nieuwe coole functie:
naamFunctie(waarde1, waarde2);
```

Simpel voorbeeld:

```
function som(x, y) {
    return x + y;
}

let s = som(1, 1);
console.log(s); // 2
```

Een functie hoeft geen naam te hebben en kan worden opgeslagen in een variabele:

```
let som = function (x, y) {
    return x + y;
}
console.log(som(1, 1)); // 2
```

Een functie kan ook zichzelf of andere functies oproepen:

```
function faculteit(n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * faculteit(n-1);
    }
}
console.log(faculteit(1)); // 1             =  1
console.log(faculteit(2)); // 1 * 2         =  2
console.log(faculteit(3)); // 1 * 2 * 3     =  6
console.log(faculteit(4)); // 1 * 2 * 3 * 4 = 24
```

Je kunt ook een functie meegeven als een parameter aan een andere functie:

```
function voerUit(func) {
    console.log("Voor func");
    func();
    console.log("Na func");
}

function uitgevoerdeFunctie() {
    console.log("Tijdens func");
}

voerUit(uitgevoerdeFunctie);
// Uitvoer:
// Voor func
```

```
// Tijdens func
// Na func
```

Opmerking: Functies worden soms ook *methoden* genoemd. Er bestaat een subtiel verschil tussen de 2, maar voor het gebruik zijn ze hetzelfde. Het verschil kan ook veranderen afhankelijk van de context waarin je spreekt. Dit maakt het moeilijk om hierop verder in te gaan zonder verwarrend over te komen. Daarom zal in de rest van dit boek overal de term functie gebruikt worden ook al is het in sommige gevallen methode (het verschil maakt eigenlijk enkel uit voor mensen die theoretisch bezig zijn met computertalen).

JavaScript gebruiken voor interactie in de browser

Elementen (Vb. `<div>...</div>`) zijn voor JavaScript objecten die je kunt manipuleren en opslaan in variabelen. De manier dat je een element kunt opvragen is door het te vragen aan een ander object `document`.

Objecten hebben de eigenschap dat ze extra informatie kunnen bevatten. Deze extra informatie kun je krijgen als een variabele of via een functie. De naam van deze variabelen en functies worden altijd voorafgegaan met de naam van de variabele waarin het object is opgeslagen, gevolgd door een punt.

Vb. Als we de functie `querySelector` willen uitvoeren op de variabele `document`:

```
document.querySelector("...");
```

In het bovenstaande geval krijg, heb je dat de parameter die je moet meegeven een selector is zoals in CSS maar nu als string (`"#id"`). De waarde die je terugkrijgt van deze functie is opnieuw een object naar het eerste element die voorkomt in je HTML die voldoet aan de selector.

```
<!-- HTML -->
...
<body>
    <div id="mijnDiv">Hallo iedereen</div>
</body>

// javascript
let mijnDivElement = document.querySelector("#mijnDiv");
let tekst = mijnDivElement.innerHTML;
console.log(tekst);
// nu gaan we de tekst veranderen
```

```
mijnDivElement.innerHTML = "Tot ziens iedereen";
```

De stijl (CSS) is op zich ook een object. Dus je kunt deze ook weer aanpassen en opslaan in een variabele. De namen van de stijlen javascript zijn de volledige namen van die in CSS. Op zich hoef je deze niet van buiten te leren: Google is je beste vriend.

```
<!-- HTML -->
...
<body>
    <div id="mijnDiv">Hallo iedereen</div>
</body>
```

```
// javascript
let mijnDivElement = document.querySelector("#mijnDiv");
let stijl = mijnDivElement.style;
stijl.color = "red"; // tekstkleur naar rood
stijl.backgroundColor = "black"; // achtergrondkleur naar zwart
```

Een belangrijke functie dat je op elementen kunt aanroepen heet `addEventListener`.

```
element.addEventListener(eventNaam, functie);
```

Deze functie laat je toe om te detecteren wanneer bepaalde dingen gebeuren. Zo kun je detecteren wanneer er op een knop wordt gedrukt (eventNaam: click) of wanneer je muis beweegt over het scherm (eventNaam: mousemove).

```
<!-- HTML -->
...
<body>
    <button id="mijnKnop">Klik mij!</button>
</body>
```

```
// javascript
let knop = document.querySelector("#mijnKnop");
knop.addEventListener("click", function (evt) {
    // evt is een object met informatie over de klik
    console.log(evt);
    console.log("Er is op mij geklikt :)")
});

let body = document.querySelector("body");
body.addEventListener("mousemove", function (evt) {
    console.log(evt);
    // coordinaten zijn altijd relatief t.o.v. het element!
```

```
    // `` is een alternatieve manier om een string te maken in javascript
    // Die dan toelaat om variabelen in te vullen als je de variabele
    // omhult met dollar{...}
    console.log(`De nieuwe positie is x: ${evt.offsetX}, y: ${evt.offsetY}`);
})
```

Opmerking: Er bestaat ook een alternatieve manier om gebeurtenissen te koppelen met javascript. Deze zijn van de vorm:

```
element.onclick = function (evt) {...};
```

Deze manier hoewel nog mogelijk, is niet meer aangeraden. Dit betekent dat je ze nog wel gaat kunnen gebruiken, maar je hebt dan geen toegang tot de nieuwe coole features of de nieuwe gebeurtenissen die worden toegevoegd aan de taal.

Opmerking: Op het internet wordt er eigenlijk weinig gesproken van een gebeurtenis (nederlandse vertaling), maar eerder van een event. Daarom wordt er vaak ook de afkorting evt gebruikt.

Buiten `document`, bestaat er nog een belangrijk object `window`. Dit object bevat informatie over de browser waarin je website zich begeeft. Je kunt hier bijvoorbeeld uit halen hoe groot het scherm is en gebeurtenissen afwachten die te maken met de browser. De belangrijkste in *load*. Deze gebeurt wanneer de volledige pagina geladen is en is dus belangrijk wanneer je *querySelector* wilt gebruiken. Dit is omdat *querySelector* enkel elementen kan terugvinden die al geladen zijn. Dus als de pagina nog niet volledig is geladen, kan het zijn dat *querySelector* je element niet vindt wat niet leuk is.

```
window.addEventListener("load", function (evt) {
    console.log("Pagina volledig geladen");
    // nu ben je zeker dat querySelector je element kan vinden
});
```

Als je een game wilt maken, heb je vaak een manier nodig om tijd mij te houden. In javascript heb je 2 manieren:

- `setInterval(functie, intervalMillis)`: Elk interval wordt de functie uitgevoerd
- `setTimeout(functie, tijdMillis)`: Na de opgegeven tijd wordt de functie 1 keer uitgevoerd en nadien nooit meer.

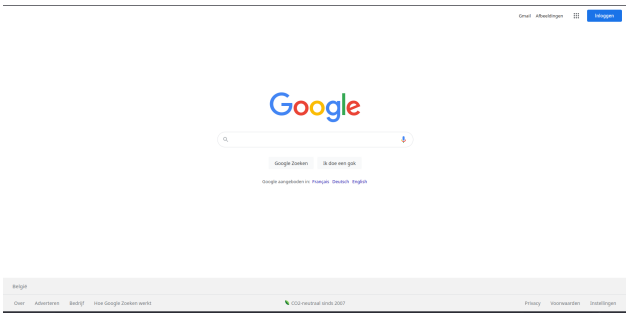
Hoofdstuk 2

Google hack

Dit hoofdstuk gaan we iets grappigs doen waarmee je familie en vrienden kan foppen. Het einddoel is om het google logo te veranderen naar iets anders grappigs.

2.1 Veranderen van logo

Als eerste ga naar www.google.be.

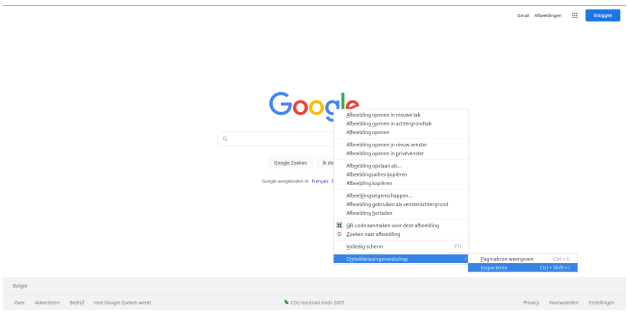


Figuur 2.1: Ga naar www.google.be

Klik met je rechter muisknop op het *google*-logo en selecteer *inspecteren*.

We krijgen dan de code te zien voor het *google*-logo. Dit ziet er uit als:

```
<img
    class="lnXdpd"
    alt="Google"
    height="92"
```



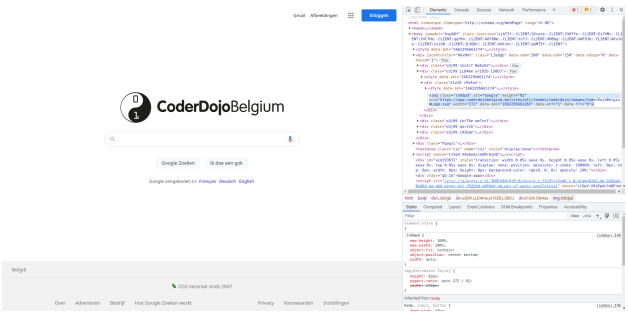
Figuur 2.2: Selecteer inspecteren van Google logo

```
    src="een zeer lange url.png"
    srcset="een nog langere url.png en dan nog iets"
    width="272"
    data-iml="1662295661263"
    data-atf="1"
    data-frt="0">
```

Verander dit naar:

```
<img
    class="lnXdpd"
    alt="Google"
    height="92"
    src="https://www.coderdojobelgium.be/sites/all/themes/coderdojo/images/Co
    width="272"
    data-iml="1662295661263"
    data-atf="1"
    data-frt="0">
```

We krijgen dan:

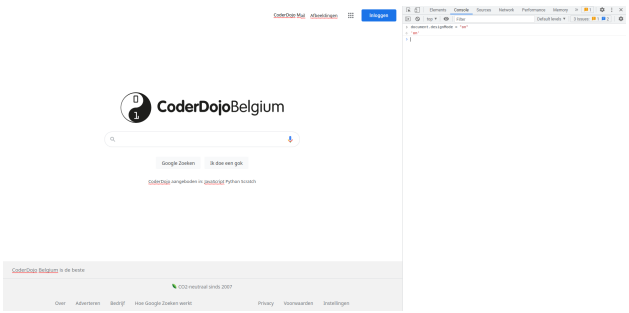


Figuur 2.3: Coderdojo search

2.2 Veranderen van andere tekst

Klik op console. Je krijgt dan een manier om direct *JavaScript* te schrijven. Schrijf nu het volgende en druk dan op *enter*:

```
document.designMode = "on"
```



Figuur 2.4: Designer mode on

Nu kun je op alle tekst klikken en deze veranderen.

2.3 Wenskaart

```
console.log('Hallo wereld');
```