> *JavaScript is the duct tape of the internet.*
> —Charlie Campbell

Het Document Object Model

Vanaf hier begint de eigenlijk uitwerking van de citaten-app. Volg mee met de filmpjes en bouw je eigen versie van deze app. Het vertrekpunt is de basiscode die in een vorig hoofdstuk al voorgesteld werd. Je kan deze startcode downloaden van Toledo. Zorg dat je console (developer tools F12) altijd open staat als je JS schrijft. Zo kan je fouten snel opsporen.

1 Browser omgeving & DOM

JS is ontwikkeld voor gebruik binnen een browser. Het is dan ook logisch dat er heel wat dingen in de taal nauw verbonden zijn met de browser. Het centrale globale object in de browser is `window`. Probeer bvb. volgende code in de console om de hoogte van het browservenster op te vragen. Maak daarna het venster iets kleiner en voer de code terug uit in de console.

```
console.log(window.innerHeight);
```

Het Document Object Model (DOM) is een voorstelling van alle inhoud van een pagina als objecten waarvan je heel wat gegevens kan opvragen of veranderen. Het object `document` is het startpunt om met het DOM te werken.

Het HTML-document wordt geladen door de browser, die er een boomvoorstelling (*tree*) van maakt. De wortel (*root*) is het element `html`. Dat element heeft twee kinderen (takken): `head` en `body`. Deze familierelaties benadrukten we reeds in beide delen HTML en CSS.

 Meer info over de DOM boom vind je op https://javascript.info/dom-nodes

In feite moeten we de uitleg in wat volgt wat vereenvoudigen. De boomstructuur bestaat namelijk uit knopen (*nodes*). De HTML-elementen waar we hierboven over spraken zijn zogenaamde *'element nodes'*. Zo is er in de code een elementknoop `h1`. Deze knoop bevat een tekstknoop (*'text node'*) met als inhoud de string 'Beroemde citaten'. Tot zover is alles eenvoudig ...

Wat de zaken wel een beetje bemoeilijkt, is dat de 'enters' (nieuwe lijn symbolen) op het einde van een regel in de HTML-code ook *nodes* zijn. Die tekstknopen zijn meestal niet erg interessant om iets mee te doen. In het vervolg van deze tekst zullen we altijd naar de elementknopen en hun inhoud verwijzen. Trouwens, alles in de HTML-code is een deel van het DOM. Een commentaar in HTML (tussen <!-- en -->) is in het DOM een *comment node* en als zodanig ook door JS te bereiken.

De developer tools van de browser tonen je deze DOM grafisch. De vervelende tekstknooppunten die overeenkomen met het nieuwe lijn symbool worden meestal niet getoond in dit overzicht. Je kan de driehoekjes voor de elementen open of dicht klikken.

 In het eerste filmpje van dit hoofdstuk starten we met het Document Object Model. We overlopen volgende topics:

- DOM bekijken in inspector (dev tools): ouders & kinderen;
- een element uit het DOM vastpakken: querySelector;
- bekijken in console (foutmeldingen);
- inhoud wijzigen: innerText;
- belang van defer attribuut.

2 Nieuw element maken en toevoegen aan je document

Het HTML-bestand is ingeladen in de browser. Als dat gebeurd is, gebruiken we JS om het DOM aan te passen. We moeten hiervoor kunnen wandelen in de boomstructuur (van ouder naar kinderen), elementen kunnen vastnemen, aanpassen, toevoegen enz.

Meer bepaald voegen we in dit deeltje een eerste citaat toe.

Volgende elementen komen in onderstaand filmpje aan bod:

- Variabelen: `const` (of `let`);
- strings met één of twee aanhalingstekens;
- formatter voegt automatisch ‘;’ toe;
- `querySelector`;
- nieuw HTML element maken (met class) via `innerHTML`;
- backticks;
- `classList.add`;
- testen in de console;
- element toevoegen aan het DOM via `appendChild`;

- modernere methode: `insertAdjacentElement`.

3 Element verwijderen uit het DOM

Soms wil je het omgekeerde: een bepaald element uit het DOM verwijderen. Zo staat er in onze HTML een paragraaf met als tekst “Er zijn momenteel nog geen citaten op deze site.”. Die vermelding mag weg want we voegden hierboven het eerste citaat toe.

In dit filmpje:

- een alternatief voor `querySelector`: `getElementByID`;
- een element verwijderen met `remove`.

4 Basisvormgeving van een citaat

Het is niet omdat deze module vooral over JS gaat, dat we het geleerde uit de eerste module vergeten. Geregeld zullen we tijdens dit grotere project wat CSS toevoegen voor een verzorgde lay-out. Vanzelfsprekend doe je dat ook voor je eigen project.

Onderstaand filmpje gaat helemaal over CSS:

- citaat als een doosje (achtergrond, padding, schaduw, afronding);
- auteur wordt cursief en rechts gezet;
- `::before` selector.

5 Tweede citaat toevoegen

We herhalen nu de vorige stappen om een nieuw citaat toe te voegen. Die herhaling geeft echter een wrange nasmaak. Het kan immers niet de bedoeling zijn om op die manier verder te werken.

In IT heeft men daar een term voor: ‘DRY’, wat staat voor ‘Don't Repeat Yourself’. Elke keer als je denkt ‘dit heb ik al eens eerder gedaan’ moet je jezelf de vraag stellen of er geen efficiëntere methode is om hetzelfde doel te bereiken.

 In volgend filmpje vind je deze topics:

- nieuw citaat toevoegen met nieuwe variabelen;
- nut van `const`;
- klein beetje CSS toevoegen om ruimte tussen articles te maken.

6 DRY → functie maken

We gaan nu een functie ‘voegCitaatToe’ maken die een citaat toevoegt. Deze functie zal later hergebruikt worden om nieuwe citaten toe te voegen. Het is een belangrijk concept in programmeren om code te hergebruiken.

 Het filmpje toont hoe we een functie ‘voegCitaatToe’ definiëren. We gebruiken hiervoor de klassieke functienotatie (later bekijken we zeker ook de ‘vette pijl’ functienotatie). Deze functie heeft vier *argumenten* nl. titel, tekst, auteur en taal (allemaal strings). De functie heeft *geen returnwaarde*. Dat wil zeggen dat ze wel een aantal dingen doet, maar dat ze geen resultaat genereert dat teruggegeven wordt. Onderaan de functiedefnitie zal je dus ook geen `return` statement vinden.

7 Footer

Meer info over het wijzigen van het document (o.a. toevoegen of verwijderen van elementen) vind je op https://javascript.info/modifying-document

Bij het maken van een footer worden volgende topics geïllustreerd:

- de methode `insertAdjacentElement`;
- een `footer` element maken;
- CSS toevoegen voor de lay-out van de footer;
- `insertAdjacentHTML` als efficiëntere oplossing.

↑ Naar top