Handboek Cursist

Javascript

Wat kan je met javascript?
Javascript is niet meer weg te denken. Moderne websites hebben vaak te maken met data die in een website verwerkt moet worden. De data staan op een server. Jouw website legt communicatie met de server om de data op te halen. De communicatie wordt gedaan met javascript. Denk aan productinformatie, voorraad van bepaalde artikelen, de prijzen van de producten. Men gebruikt javascript om data op te halen van de server, te valideren en te manipuleren. Bijvoorbeeld, Als je de voorraad van een product ophaalt en de voorraad is op, dan wil je graag op de website tonen dat het uitverkocht is. Met javascript kun je valideren dat het voorraad op is en vervolgens kan aangeven dat er een label “uitverkocht” op de pagina moet staan.

Met javascript zou je ook de content van de html wijzigen of de style van je css aanpassen.

Hoe werkt het ?
Javascript kennen is nodig om bepaalde logica uit te voeren. Een programmeertaal zoals javascript is een human readable taal waarmee jij als ontwikkelaar de code kan uitschrijven. Net als bij Nederlands heb je met een javascript regels die je moet hanteren. Door het goed toepassen van deze regels zult de computer dit begrijpen en de acties uitvoeren die jij hebt geprogrammeerd. Ook zijn er standaard regels die toegepast voor een betere leesbaarheid van de code.

Een voorbeeld van javascript regels:

```
class Car {
    constructor(height, width) {
      this.height = height;
      this.width = width;
    }
}
const car = new Car(20, 20);
console.log(car.height);
console.log(car.width);
```

Hierboven zie je een stukje code waar de regels goed toegepast zijn. Javascript is een case-sensitive programmeertaal. In dit voorbeeld is Car (met hoofdletter) en car niet hetzelfde. De computer zal het anders interpreteren. Verder zie je dat er indentations worden gebruikt. Dat zijn spaties die je gebruikt om jouw code te structureren. Ook zie je dat aan het einde van de regel een bracket of een semi colon gebruikt worden. Doe je dit niet dan overschrijd je de regels en breekt jouw code.

Javascript in ontwikkeling
Javascript wordt ook wel ECMAScript genoemd. In de IT gaat technologie razendsnel. Websites die enkele jaren geleden zijn gemaakt zijn al verouderd. Je loopt dan achter qua

technologie en als je dit niet bijstuurt kan dat gevaar opleveren voor bedrijven waar IT de core business is. Dat geldt ook met programmeertalen zoals Javascript. Javascript van 10 jaar geleden ziet er anders uit dan tegenwoordig. De taal is in ontwikkeling en zal continue worden verbeterd. Hierdoor ontstaat dat bepaalde functionaliteit op verschillende manieren de code uitgeschreven kan worden. Dit maakt het voor ontwikkelaars niet makkelijk, want hierdoor ontstaat er discussies welke code beter is.

```
var numbers = [1, 2, 3, 4, 5];

// example1
for(var number of numbers) {
    console.log(number);
}

// example2
numbers.map(n => {
    console.log(n);
})
```

Hierboven zie je example1 en example2 waarbij de getallen van de lijst worden uitgeprint. Qua functionaliteit doen beide voorbeelden exact hetzelfde. Example1 is typisch een voorbeeld dat vroeger heel vaak wordt geschreven. Nadat de taal verbeterd was is Example2 nu heel populair om te schrijven waardoor Example1 eigenlijk niet meer wordt gebruikt. Hoe bepaal je nu welke code je kan gebruiken? Dat is afhankelijk of de browser het support. Elke browser heeft een bepaalde versie. En elke versie van de browser ondersteunt bepaalde ECMAscript versie. Het kan zijn dat je een oudere versie van de browser gebruikt waarbij de browser de code van “example2” niet begrijpt. Hierdoor ben je genoodzaakt je browser te updaten, zodat de browser de “nieuwere” code begrijpt of de code omschrijven naar het voorbeeld zoals “example1”. Tegenwoordig ondersteunen meeste browsers de nieuwste ECMAScript versie. Ondanks dat is het wel iets om over na te denken, omdat je de zekerheid niet hebt dat browsers al jouw code ondersteunt. Om zeker van te zijn dat je code werkt zul je op verschillende browsers en versies testen of het werkt.

Javascript die in je browser uitgevoerd wordt noemen we ook wel client-side rendering. Client-side rendering houdt in dat het op de computer waar jij achter zit wordt uitgevoerd. Server-side rendering houdt in dat de code die je hebt geschreven op een andere computer wordt uitgevoerd. Een server is niks anders dan een computer die vaak fysiek niet in jouw huis of kantoor staat, maar bij een datawarehouse.

Leerdoelen

- Javascript syntax
- Browser console
- Debugging

De syntaxis van javascript is essentieel om te beheersen. Dit vormt de basis van jouw programmeerkennis. Door dit goed te beheersen zul je andere programmeertalen makkelijker kunnen oppakken. Er zijn namelijk onderdelen die in andere programmeertalen ook in voorkomen. Deze onderdelen zullen wij stapsgewijs gaan behandelen. Hiervoor hebben wij opdrachten voor die jullie kunnen maken.

Naast het leren van Javascript is het ook belangrijk om aan te leren hoe jij je code kan debuggen. Code zul je niet in een keer goed kunnen schrijven. Fouten maken kan je niet voorkomen. Van fouten kun je leren en verbeteren. Met debuggen kun je stap voor stap jouw code uitvoeren en zien wat er gebeurd. Daar zullen wij ook tijd aan besteden.

Aan het einde van dit onderdeel ken je de basis van Javascript en kun je op je eigen manier jouw code schrijven en de fouten opsporen.

Javascript introductie

Het zal in het begin enigszins moeilijk te bevatten waarom het belangrijk is om de verschillende onderdelen van javascript te leren. Dit komt omdat de onderdelen samenhang met elkaar hebben. De samenhang tussen de verschillende onderdelen vormen functionaliteiten van jouw website. Wij behandelen de verschillende onderdelen een voor een zodat er een goede focus is op het leren. Zodra alle essentiële onderdelen geweest zijn zullen wij de verschillende onderdelen combineren en zul je wat complexere applicatie maken zoals een rekenmachine en wordt het enigszins ook duidelijker waar javascript voor dient.

Om mee te beginnen gaan wij het hebben over data types en variables. Tijdens programmeren zul je merken dat data types en variables heel vaak terugkomen. Het is essentieel om te begrijpen wat de verschillende datatypes zijn en hoe variables werken.

Leerdoelen

- Data types
- built in methods
- Variables

Data types

In javascript zijn negen data types.

Er zijn zes primitive data types:

- undefined. De waarde undefined betekent dat er geen waarde aan de variable is toegekend.
- Boolean. De waarde is altijd true of false
- String. De waarde is altijd een tekst. Een tekst staat altijd tussen dubbele quotes (“”). Bijvoorbeeld, “Dit is een tekst”
- Number. De waarde van een Number is altijd een getal.
- BigInt. Een BigInt lijkt heel veel op Number. De waarde van BigInt is een getal en eindigt op n. Bijvoorbeeld 3n. De grootste verschil tussen BigInt en Number is dat BigInt oneindig getallen kan opslaan. Dit is niet nodig om te onthouden.
- Symbol. Symbol wordt gebruikt om de waarde een unieke key van te maken . Wordt heel weinig gebruikt en is niet nodig om te onthouden.

Daarnaast heb je ook nog:

- null. De waarde is een null als de waarde ook expliciet op null gezet wordt. Null wordt vaak gebruikt om aan te duiden dat de waarde niet bestaat.
- Object. Een object is een groepering van variables en functies. Je kan het zien dat variables in een object de eigenschappen zijn en functies het gedrag van dat object. Objecten zul je vaak tegenkomen, omdat je ook objecten kunt maken.

```
class Person {
  constructor(fname, lname) {
```

```
        this.firstName = fname;
        this.lastName = lname;
    }

    getName() {
        return this.firstName + " " + this.lastName;
    }
}
const person1 = new Person('Kim Sing', 'Cheng');
const person2 = new Person('Jonathan', 'lee');
console.log(person1.getName());
console.log(person2.getName());
```

- Function. Een functie is een set aan regels of acties die uitgevoerd worden. Aan een functie kun je een leesbare naam geven. De naam representeert de actie die uitgevoerd wordt. Bijvoorbeeld:

```
function sum(number1, number2) {
    return number1 + number2;
}

console.log(sum(3, 4)) // 7
```

Variables

Variables houdt de waardes vast van een bepaalde data type. Variables kent drie varianten:

- **Var**. var is de voorganger van let en wordt door oudere browsers ondersteunt. Tegenwoordig wordt var niet meer gebruikt en is het vervangen door let
- **Let**. let is de nieuwe manier om waarde vast te houden. Wanneer je let gebruikt is de waarde mutable. Dat wil zeggen dat je de waarde van de variable kan aanpassen.
- **Const**. Const wordt ook gebruikt om waarde vast te houden. Het verschil tussen let en const is dat const immutable is. Dat wil zeggen dat wanneer je een waarde hebt toegekend aan een variable dat het dan ook niet meer te wijzigen is. Het niet toewijzen van een waarde aan een variable zal een error geven.

built in methods

Methods zijn in feite functies. Functies worden later behandeld. Om alvast een voorproefje te krijgen wat functies zijn, zijn er built in methods die je kunt gebruiken om bepaalde actie mee uit te voeren. Built in methods houdt in dat er een set aan functies die al voorgedefinieerd zijn die je kunt gebruiken. Een bekende voorbeeld met built in methods is de String. String is een object. Dat kun je herkennen doordat String met een hoofdletter geschreven is. Het is een geschreven regel dat objecten met een hoofdletter begint. Daardoor kun je het herkennen dat het een object is. Binnen objecten heb je functies die je kunt gebruiken. Later zullen wij onze eigen objecten maken met onze eigen functies. Voor nu kunnen we kijken naar de String object en welke functies er zijn. Een String bevat een lijst met built in

methods. Wij zullen enkele voorbeelden laten zien. De rest kun je vinden op: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

```
let text = "abcdefg";

console.log(text); // abcdefg.
console.log(text.charAt(2)); // c
console.log(text.substr(0, 3)); // abc
console.log(text.replace("c", "d")); // abddefg
```

Gebruik bovenstaande code om uit te leggen hoe built in methods werken. Het is belangrijk om ook te snappen hoe de documentatie werken. De documentatie legt namelijk uit hoe je de built in methods kan toepassen.

Opdrachten

Voor de opdrachten kunnen deelnemers VisualStudio code gebruiken om de opdrachten uit te voeren.

Deze opdrachten zijn bedoeld om hands on ervaring op te bouwen.

Prerequisites
Maak een nieuwe html bestand aan in VSCode. Copy de volgende code in jouw html bestand.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

    // hier kun je de antwoorden van de vragen plaatsen.
  <script>
    // hier kun je coderen
  </script>
</body>
</html>
```

Datatypes

Opdrachten over undefined en getallen aan variable toekennen.

Opdracht 1:

1. Maak een variable aan met let en noem het “getal1”.
2. Ken geen waarde toe aan de variable.
3. Print vervolgens de variable met console.log(“getal1 ” + getal1);.

Open dit html bestand in jouw browser en kijk in je console. Wat zie je dan?

Opdracht 2:

1. Maak een variable aan met const en noem het getal2.
2. Ken geen waarde toe aan de variable.
3. Print vervolgens de variable met console.log(“getal2 ” + getal2);.

Open dit html bestand in jouw browser en kijk in je console. Wat zie je dan?

4. Maak een variable aan met const en noem het getal2.
5. Ken een Number toe aan de variable.
6. Print vervolgens de variable met console.log(“getal2 ” + getal2);.

Opdracht 3:

1. Maak een let variable en noem het getal3. Ken gelijk een Number 3 aan toe. Hoe programmeer je dat ?
2. Print vervolgens de variable met console.log(“getal3 ” + getal3);.

Open dit html bestand in jouw browser en kijk in je console. Je zult “getal3 3” in console zien.

Opdracht 4:

1. Hergebruik de variable "getal3 en wijzig de waarde van de variable getal3 naar 5.
2. Print vervolgens de variable met console.log(“getal3 ” + getal3);

Open dit html bestand in jouw browser en kijk in je console. Je zult “getal3 5” in console zien

Opdrachten over de verschillende data types herkennen

Opdracht 5:

1. Maak een let variable en noem het datatype.
2. Ken de waarde true aan de variable.
3. Print vervolgens de variable met console.log(“datatype ” + typeof datatype);

Open dit html bestand in jouw browser en kijk in je console. Je zult “datatype boolean” in console zien.

4. Wat voor type is de waarde false?

Opdracht 6:

1. Herhaal opdracht 5 en geef antwoord op de volgende vragen.

Hoe herken je dat een waarde String is? Geef ook een voorbeeld.

2. Hoe herken je dat een waarde Number is? Geef ook een voorbeeld.

3. Hoe herken je dat een waarde BigInt is? Geef ook een voorbeeld.

4. Hoe herken je dat een waarde Symbol is? Geef ook een voorbeeld.

Opdrachten wat je met Numbers kan doen.

Opdracht 7:

1. Maak twee variables aan met twee verschillende Numbers. Geeft de variables een logische naam waaraan je kunt afleiden wat het inhoudt.
2. Eerste variable heeft de waarde 6.
3. Tweede variable heeft de waarde 8.

Tel de twee waardes bij elkaar op. Print het antwoord in jouw browser console. Het antwoordt zal 14 moeten zijn.

Trek de tweede waarde af van de eerste waarde. Print het antwoord in jouw browser console. Het antwoordt zal 2 moeten zijn.

Vermenigvuldig de twee waardes. Print het antwoord in jouw browser console. Het antwoordt zal 48 moeten zijn.

Deel de twee waarde met de eerste waarde. Print het antwoord in jouw browser console. Het antwoordt zal 1.33 moeten zijn.

Opdracht 8:

1. Wat is het antwoord van de volgende som: 3 + 5 * 10 / 2?
2. Print het antwoord in jouw browser console.

Open dit html bestand in jouw browser en kijk in je console.

Opdracht 9:

Voor deze opdracht is het handig om je in te lezen over increment en decrement.

1. Copy de volgende code in jouw VSCode:

```
let number = 3;
number++;
```

2. Print uitkomst in jouw browser console.

Wat gebeurt de uitkomst?

3. Wijzig number++ naar number--.
4. Wat is de uitkomst van de som ?

Wat gebeurt is de uitkomst?

5. Copy de volgende code in jouw VSCode:

```
let number = 3;
const sum = number++ + 2;
console.log(sum);
```

6. Wat is de uitkomst van van de som ?

Leg jouw antwoord uit.

7. Copy de volgende code in jouw VSCode:

```
let number = 3;
const sum = ++number + 2;
console.log(sum);
```

8. Wat is de uitkomst van van de som ?

Opdracht 10:

1. Copy de volgende code in jouw VSCode:

```
    let number = 3;
number += 4;
console.log(number);
```

2. Wat is de uitkomst van van de som ?
3. Hoe zou je het anders kunnen schrijven en dezelfde uitkomst hebben ?

Geef een **antwoord**, **geschreven code** en **uitleg**.

Opdracht 11:

1. Copy de volgende code in jouw VSCode:

```
let number = 3;
number += 4;
number -= 2;
number /= 5;
number *= 4;
console.log(number);
```

2. Wat is de uitkomst ?

Opdrachten wat je met String kan doen.

Opdracht 12:

1. Maak twee variables aan met twee verschillende Strings. Geef de variables een logische naam waaraan je kunt afleiden wat het inhoudt.
2. Eerste variable heeft de waarde “Voornaam”.
3. Tweede variable heeft de waarde “Achternaam”.
4. Print vervolgens de variables met console.log(<eerste variable naam> + <tweede variable naam>);

Wat merk je op t.o.v het uitprinten van numbers?

Opdracht 13:

1. Maak een String variable aan met de waarde “3”.
2. Maak een Number variable aan met de waarde 3.
3. Maak een variable aan die heet combinedText en voeg de eerste en de tweede variable samen en ken de samengevoegde waarde toe aan de variable combinedText.
4. Print vervolgens de combinedText in jou console.

Wat merk je op?

Opdracht 14:

1. Een String kun je ook aanduiden met single quote (‘). Wanneer zou je een String met single quote gebruiken?
2. Maak een voorbeeld code waarbij je een single quote zou gebruiken.

Opdracht 15

1. Copy de volgende code in jouw VSCode:

```
let text = "text";
text += "another text";
console.log(text);
```

2. Wat wordt er geprint ?
3. Hoe zou je het anders kunnen schrijven en dezelfde uitkomst hebben ?

Opdrachten met built in methods

Opdracht 16:

Voor deze opdracht is het handig om in te lezen over methods van String. Aan de linkerkant staat een menubalk. Onder de categorie methods staan een lijst met functies die je met een String kan aanroepen.

Maak een eigen voorbeeld van de volgende functies en licht toe wat de functie doet in het blok:

1. charAt()
2. concat()
3. slice()
4. split()
5. substr()

Conditions

Vandaag gaan wij het hebben over conditions. Als wij nu niet kijken naar programmeren, maar kijken naar hedendaagse gebeurtenissen, dan hebben jullie vast ook wel eens keuzes moeten maken bij een bepaalde gebeurtenis. De keuze die je maakt is gebaseerd op bepaalde voorwaardes en gevolgen die er aan hangt.

Een voorbeeld waar je een keuze zou kunnen maken is wanneer je van plaats A naar plaats B reist. Stel je wilt van A'dam west naar A'dam oost reizen. Dan kan je afvragen of je met de auto, de fiets, de scooter of de tram pakt? Er zijn overwegingen die je kunt maken. Bijvoorbeeld, je kunt overwegen om met de auto te gaan, omdat het dan ongeveer 25 min rijden is, maar daarentegen heb je te maken met. Of je gaat met de fiets, dan kost het jou 45 min om te fietsen en zijn de vervoerskosten gratis. Of kies je de middenweg en ga je met de scooter? Als laatste optie zou je ook nog de tram kunnen pakken. Het kost je dan ongeveer 3 euro, maar dan duurt het ongeveer 60 min om naar jouw bestemming te komen.

Welke vervoersmiddel je pakt is afhankelijk van de conditie. Als het belangrijk is dat je binnen nu en 30 min op jouw bestemming moet zijn, dan zal je de auto moeten nemen. Als je over een uur kan zijn, dan kan je overwegen om de fiets of de tram te pakken. Als je over een uur kan zijn en je wilt geen kosten maken, dan neem je de fiets.

Deze condities kun je programmeren in javascript. Je hebt met condities te maken met voorwaardes en acties. Als je aan bepaalde voorwaarde voldoet, dan kan je bepaalde actie uitvoeren. Met het voorbeeld die ik net noemde is de voorwaarde verbonden aan tijd en kosten. En de actie is het daadwerkelijk reizen naar jouw bestemming met een vervoermiddel.

Leerdoelen

- If, else if, else conditions
- Switch conditions
- Ternary conditions

Opdrachten

Voor de opdrachten kunnen deelnemers VisualStudio code gebruiken om de opdrachten uit te voeren.

Deze opdrachten zijn bedoeld om hands on ervaring op te bouwen.

Prerequisites
Maak een nieuwe html bestand aan in VSCode. Copy de volgende code in jouw html bestand.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    // hier kun je coderen
  </script>
</body>
</html>
```

Opdrachten over if statements met comparison operators

Opdracht 1:

Copy de volgende code in jouw body element:

```
  <input type="number" name="number" id="number">
  <button id="btn" onclick="checkNumber()">Button</button>

  <div>
    <span id="label"></span>
  </div>
  <script>
    function checkNumber() {
                  // hier kun je coderen
    }
```

```
</script>
```

Hierboven zie je een input field, een button en een label.

- Het input field kun je een getal mee invoeren.
- De button roept de checkNumber functie aan.
- De label wordt gebruikt om een tekst te tonen aan de hand van een logica die jij gaat programmeren.

Programmeer de volgende logica in de checkNumber functie.

- Als het ingevoerde getal **is gelijk aan** 3, dan toon je op het label: “Het ingevoerde getal is gelijk aan 3”.

Opdracht 2:

Deze opdracht is een uitbreiding van opdracht 1.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is groter dan** 4, dan toon je op het label: “Het ingevoerde getal is groter dan 4”.

Opdracht 3:

Deze opdracht is een uitbreiding van opdracht 1.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is gelijk aan** en **is groter dan** 11, dan toon je op het label: “Het ingevoerde getal is gelijk aan 11 of groter dan 11.”.

Opdracht 4:

Deze opdracht is een uitbreiding van opdracht 1.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is kleiner dan** 3, dan toon je op het label: “Het ingevoerde getal is kleiner dan 3”.

Opdracht 5:

We hebben nu verschillende comparison operators gehad. Wat is het verschil tussen < en <= ?

Opdracht 6:

We hebben nog een comparison operator niet gehad, namelijk **!** comparison. Wat doet een **!** comparison ? Maak een voorbeeld van een **!** comparison

Opdracht 7:

Wanneer je een getal hebt ingevoerd en bij de label verschijnt er een tekst, dan blijft de tekst voor altijd staan tenzij je de tekst van de label expliciet weghaalt. Schrijf de code van opdracht 1 t/m 4 zo dat als de conditie niet waar is dat de tekst van de label verwijdert wordt.

Opdracht 8:

Wat is een strict equality comparison ?

Opdrachten over if statements met logical operators

Opdracht 9:

Maak een nieuwe html bestand aan en copy de volgende code in jouw bestand. Dit is hetzelfde als het begin bestand van opdracht 1.:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>

  <input type="number" name="number" id="number">
  <button id="btn" onclick="checkNumber()">Button</button>

  <div>
    <span id="label"></span>
  </div>
  <script>
    function checkNumber() {
        // hier kun je coderen
    }
  </script>
```

```
</body>

</html>
```

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is groter dan** 5 **en is kleiner dan** 10, dan toon je op het label: “Het ingevoerde getal is tussen 5 en 10”.

Opdracht 10:

Deze opdracht is een uitbreiding van opdracht 7.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is gelijk aan** 13 **of is groter dan** 13 en het ingevoerde getal **is gelijk aan** 20 of **kleiner dan** 20, dan toon je op het label: “Het ingevoerde getal is gelijk aan 11 of groter dan 11 en is gelijk aan 20 of kleiner dan 20.”.

Opdracht 11:

Deze opdracht is een uitbreiding van opdracht 7.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **is gelijk aan** 21 **of** het ingevoerde getal **is gelijk aan** 23 , dan toon je op het label: “Het ingevoerde getal is gelijk aan 21 of is gelijk aan 23.”.

Opdracht 12:

Deze opdracht is een uitbreiding van opdracht 7.

Programmeer de volgende logica in de checkNumber functie:

- Als het ingevoerde getal **groter is dan** 30 **en kleiner is dan** 35 **of** als het ingevoerde **getal groter is dan** 40 **of kleiner is dan** 45, toon dan op het label: “het getal is tussen 30 en 35 of het getal is tussen 40 en 45.”

Opdracht 13:

Maak een nieuwe html bestand aan en copy de volgende code naar jouw bestand:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
```

```
  <h1>Translate months</h1>
  <input type="number" name="number" id="number">
  <button onclick="translateMonth()">translate</button>
  <div id="translatedMonth"></div>

  <script>

    function translateMonth() {
                    // Hier kun je starten met coderen.

    }
  </script>
</body>
</html>
```

Deze applicatie moet het volgende kunnen doen:

- Het vertalen van getallen naar maanden. Bijvoorbeeld, als je het getal 1 invoert dan toon je op het scherm “Januari”.
    - 1 => January.
    - 2 => February.
    - 3 => March.
    - 4 => April.
    - 5 => May.
    - 6 => June.
    - 7 => Juli.
    - 8 => August.
    - 9 => September.
    - 10 => Oktober.
    - 11 => November.
    - 12 => December.

Opdracht 14:

Maak opdracht 13 opnieuw, maar gebruik nu **Switch** statement om dezelfde resultaten te maken.

Opdracht 15: Guess the word!

Maak een nieuwe html bestand aan en copy de volgende code naar jouw bestand:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<style>
    .words button {
        position: relative;
        border: 1px solid black;
        border-radius: .5em;
        background-color: white;
        padding: .5em;
        cursor: pointer;
        transition: background-color .3s ease-in-out;
        z-index: 1;
    }

    .words button:focus {
        outline: none;
        background-color: grey;
    }

    .words .fg {
        position: absolute;
        content: '';
        display:flex;
        justify-content: center;
        align-items: center;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        border-radius: inherit;
        background-color: white;
        z-index: 2;
        transition: width .3s ease-out;
        overflow: hidden;
    }

    .words button:hover .fg {
        width: 0;
        z-index: 1;
    }
```

```
    .words button:focus .fg {
      width: 0;
    }

    .answer {
      font-size: 2em;
      font-weight: 900;
    }
  </style>
</head>

<body>
  <h1>Guess the word</h1>
  <section>
    <div class="words">
      <button onclick="guess(this)"><div
class="fg">Show</div><span>Together</span></button>
      <button onclick="guess(this)"><div
class="fg">Show</div><span>Teamwork</span></button>
      <button onclick="guess(this)"><div class="fg">Show</div><span>Scrum</span></button>
      <button onclick="guess(this)"><div
class="fg">Show</div><span>Individual</span></button>
      <button onclick="guess(this)"><div
class="fg">Show</div><span>Moments</span></button>
      <button onclick="guess(this)"><div class="fg">Show</div><span>Like</span></button>
      <button onclick="guess(this)"><div class="fg">Show</div><span>Dislike</span></button>
      <button onclick="guess(this)"><div class="fg">Show</div><span>Money</span></button>
      <button onclick="guess(this)"><div
class="fg">Show</div><span>Important</span></button>
    </div>
  </section>

  <div>
    <span>The number of wrong guessing: <span id="count">0</span></span>
    <p id="label"></p>
  </div>
  <script>
    const words = ["Together", "Teamwork", "Scrum", "Individual", "Moments", "Like", "Dislike",
"Money", "Important"];
```

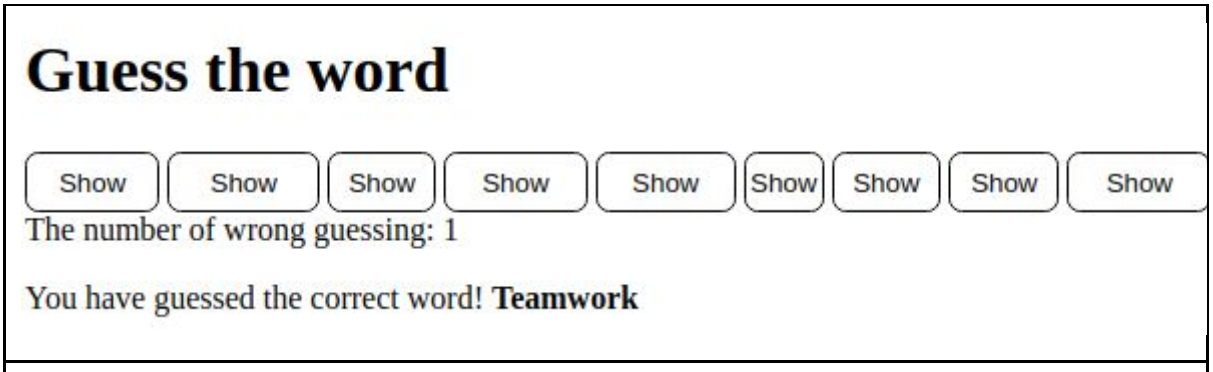
```
    const guessingWord = words[Math.round(Math.random() * 8)];

    function guess(obj) {
    }

  </script>
</body>

</html>
```

Analyseer de code en probeer te begrijpen wat er staat.

Deze applicatie moet het volgende kunnen doen:

- Het bijhouden van het aantal pogingen voor het raden van het woord.
- Na 3 keer raden is het game over en moet het volgende tekst op het scherm weergegeven worden: “Game Over. The correct word is <het woord dat geraden moest worden>”. De p.label kun je hiervoor gebruiken om de tekst op het scherm weer te geven.
- Als het woord goed geraden is dan toon je het volgende tekst op het scherm: ““You have guessed the correct word! <het woord dat geraden moest worden>.

Zie hieronder twee voorbeelden van de applicatie:



Dag 3 - Functions

Functions zijn coding blocks die uitgevoerd worden. Het idee achter functies zijn het uitvoeren van specifieke taken. Er zijn aantal regels die je moet kennen om de kracht uit functies uit te halen. Door goed gebruik te maken van functies is je code veel leesbaarder.

Wat is belangrijk om mee te nemen bij het schrijven van een function

1. *Naam*: Elke function die je schrijft heeft een naam. Deze naam geeft aan welke actie er uitgevoerd wordt. Door een goede naam aan te geven hoef je niet in code te verdiepen om te begrijpen wat het doet. Code wat werkt wil je het liefst zo weinig tijd aan besteden. Code begrijpen kan namelijk tijdrovend zijn, omdat je bij elke line of code erover gaat nadenken wat het doet. Code wordt geschreven met een bepaalde doel. Jij als ontwikkelaar wilt eigenlijk de doel van de code achterhalen. Als de code complex is kan het uren duren voordat je de code begrijpt. Door goede namen te geven begrijp je de code veel sneller.
2. *Parameters*: Je kunt values meegeven aan de function. De functie kan vervolgens de value gebruiken om bepaalde logica uit te voeren.
3. *Return*: Een function kun je aanroepen. Degene die het aanroept kan ook een waarde terugkrijgen van de function. De waarde is vaak de resultaat van de taak die de function heeft uitgevoerd.

Leerdoelen

- Functions
    - Function names
    - Parameters
    - Arrow functions
    - Returns

Opdrachten

Voor de opdrachten kunnen deelnemers VisualStudio code gebruiken om de opdrachten uit te voeren.

Deze opdrachten zijn bedoeld om hands on ervaring op te bouwen.

Prerequisites
Maak een nieuwe html bestand aan in VSCode. Copy de volgende code in jouw html bestand.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Functions</title>
</head>
<body>
    <p id="answer"></p>
    <button onclick="answer()">click</button>
    <script>

        function answer() {
            // hier kun je coderen
        }

        // Plaats hieronder alle antwoorden van de opdrachten.
    </script>
</body>
</html>
```

Opdracht 1:

Schrijf een functie die de tekst “hello world” in de console uitprint. Geef een goede functie naam ervoor.

Opdracht 2:

Schrijf een functie waarbij je een tekst mee kan geven aan de functie. Print vervolgens de tekst uit in de console.

Opdracht 3:

Schrijf een functie waarbij je een getal mee kan geven aan de functie. Print vervolgens het getal uit in de console.

Opdracht 4:

Schrijf een functie die twee getallen met elkaar vermenigvuldigt en geef de waarde van de vermenigvuldiging terug. Geef een goede functie naam ervoor.

Opdracht 5:

Schrijf een functie waarbij je een naam kan meegeven aan de functie. De functie moet vervolgens controleren of de naam een bekende is.

De namen die bekend zijn:

- John
- Olly
- Mo

- Hendrik
- Tony

Als het een bekende is, dan toon je het volgende tekst op het scherm: “Dit is een bekende van ons: <naam die je aan de functie hebt meegegeven>”.
Als het **niet** een bekende is, dan toon je het volgende tekst op het scherm “Onbekend in ons systeem!”

We gaan de applicatie uitbreiden! Je hebt een functie geschreven waarbij je zelf een naam meegeeft aan de functie. Nu willen we dat de naam random gekozen wordt. We gaan gebruik maken van Math.random() en Math.round() functie. Lees je goed in wat beide functies doet.

Schrijf nu een functie die een random naam teruggeeft. Math.random() geeft een getal terug. Zorg ervoor dat getal en naam samenhang hebben. Bijvoorbeeld getal 1 geeft aan dat het “John” is. Hieronder zie je de getallen en de namen die samenhang met elkaar hebben.

- 1 = John
- 2 = Oly
- 3 = Mo
- 4 = Hendrik
- 5 = Tony
- 6 = Irus
- 7 = Jan
- 8 = Peter

Maak een functie die een random naam teruggeeft. Deze naam geef je vervolgens door aan de functie die je voorheen hebt geschreven om te controleren of de naam een bekende is van ons.

Opdracht 6: Arrow functions

De functies die kunnen ook met arrow functions geschreven worden. Herschrijf opdracht 5 zodat arrow functions gebruikt wordt.

Opdracht 7:

Hoe geef je 2 of meer parameters mee aan een functie? Geef een voorbeeld.
Wat zijn rest parameters? Geef een voorbeeld.
Wat is het verschil tussen rest parameters en expliciete parameters die je aangeeft bij een functie?

Opdracht 8: Student grades!

Maak een nieuwe HTML bestand aan.

Wij gaan een applicatie maken om de cijfers van de studenten bij te houden. De leraar wilt op een pagina de cijfers kunnen invullen en controleren of de studenten over kunnen gaan naar de volgende klas. Met deze applicatie zorg je ervoor dat de leraar niet handmatig hoeft te

berekenen wat de gemiddelde cijfers zijn per student. Hiermee heeft de leraar een compleet overzicht van de studenten. Dat scheelt tijd voor de leraar!

Wat moet de applicatie kunnen?

- Voor de vakken wiskunde, natuurkunde, nederlands en engels een tabel met studenten waarbij de cijfers bijgehouden worden.
- Als de cijfer lager dan 5.5 is, dan is het veld rood.
- Als de cijfer hoger of gelijk aan 5.5 is, dan is het veld groen.
- Als de cijfer hoger of gelijk aan 7.5 is, dan is het veld donker groen.
- Als je op de cijfer clickt kan de cijfer aangepast worden. Zodra de cijfer aangepast is, dan wordt er opnieuw gekeken welke kleur de veld heeft (rood, groen of donker groen)
- Naast de vakken heb je nog een tabel waarmee je de gemiddelde cijfer berekend van de vakken per student. De tabel houdt bij wat de gemiddelde cijfer is van de talen (nederlands en engels) en de gemiddelde cijfer van de bètavakken (wiskunde en natuurkunde).
- Als de gemiddelde van de talen en de gemiddelde van de bètavakken gemiddeld boven de 6.5 ligt, dan gaat de student over naar volgende klas.

Zie hieronder de resultaat wat het moet worden:

Wiskunde

Naam	Cijfer
Willem	3
Jan	7
Kim Sing	10
Samantha	4
Joyce	5.5
Gareth	1

Natuurkunde

Naam	Cijfer
Willem	7
Jan	5
Kim Sing	8
Samantha	6
Joyce	9
Gareth	5.5

Nederlands

Naam	Cijfer
Willem	7
Jan	7
Kim Sing	4
Samantha	8
Joyce	8.5
Gareth	4

Engels

Naam	Cijfer
Willem	6
Jan	6
Kim Sing	5.5
Samantha	4.3
Joyce	5.5
Gareth	8

Resultaat gemiddeld

Naam	Talen	Beta vakken	Volgende klas? > 6.5
Willem	6.5	5	No
Jan	6.5	6	No
Kim Sing	4.5	9	Yes
Samantha	6	5	No
Joyce	6.5	7	Yes
Gareth	6	3	No

Scopes

Scopes geeft de context aan waar de variables gedeclareerd worden. Dit geeft de toegankelijkheid van de variables aan. Afhankelijk van de scope kan je dus toegang krijgen tot de waarde van de variable.

Leerdoelen

- Global scopes
- Block scopes

Opdracht 1:

Kijkend naar de volgende code:

```
function printCar() {
    const carBrand = "Volvo";
}

console.log(carBrand);
```

Wat wordt er in de console gelogt? Wat voor scoping heb je hier te maken?

Opdracht 2:

Kijkend naar de volgende code:

```
let carName;
function printCar() {
    carBrand = "Volvo";
}

console.log(carBrand);
```

Wat wordt er in de console gelogt?

Opdracht 3:

Kijkend naar de volgende code:

```
function printCar() {
    carBrand = "Volvo";
}

printCar();
console.log(carBrand);
```

Wat wordt er in de console gelogt?

Opdracht 4:

Kijkend naar de volgende code:

```
function printCar() {
    carBrand = "Volvo";

    return function(type) {
        console.log(carBrand + " " + type);
    }
}

const car = printCar();
car("v40");
```

Wat wordt er in de console gelogt? Hoe is dit anders voorgaande opdrachten? Lees je meer over closures:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures

Arrays

Voorheen hebben jullie gehad hoe Strings, Booleans, numbers etc werken en hoe je een waarde toekent aan de variable met een bepaalde data type. Arrays is ook een data type. Arrays is een collectie met elementen. Het houdt een lijst met waardes bij. De waardes van de array kunnen van verschillende data types zijn.

Hieronder zie je stapsgewijs hoe je een array kan aanmaken, hoe je de array kan vullen en hoe je een value kan benaderen. Om de waarde uit de array te lezen gebruiken we index. De index geeft de positie van de array aan. De telling van een array begint bij 0. Anders dan wat wij gewend zijn om de telling bij 1 te beginnen. Neem je tijd om te bestuderen hoe arrays werkt.

```
<script>

    // 1. Aan maken van een array. met [] geef je aan dat je een
lege lijst wilt maken.
    let products = [];

    // 2. Toevoegen van producten in de array
    products = ["Schoenen", "Sjaal", "Handschoenen", "Muts"];

    // 3. Een waarde benaderen in de array.
    console.log(products[0]);
```

```
</script>
```

We hebben nu een array gemaakt met String als values. Je zou ook andere data types in kunnen opslaan..

```
<script>

    // 1. Aan maken van een array. met [] geef je aan dat je een
lege lijst wilt maken.
    let randomNumbers = [];

    // 2. Toevoegen van producten in de array
    randomNumbers = [1, 4, 5, 8, 9];

    // 3. Een waarde benaderen in de array.
    console.log(randomNumbers[4]);

</script>
```

Leerdoelen

- Aanmaken van arrays
- Arrays functies gebruiken
- De waardes van een array uitlezen
- De indexes

“Opdracht 1 t/m 9 horen bij elkaar. Maak deze opdrachten ook in 1 js bestand.”

Opdracht 1:

Maak een array met 10 getallen die willekeurig in een array staan. Print de array uit in jouw console.

Opdracht 2:

Maak een array met 5 verschillende fruitsoorten. Print de array uit in jouw console.

Opdracht 3:

Op welke index staat “Appel”. Gebruik de index waarde om “Appel” in jouw console uit te printen.

Opdracht 4:

Op welke index staat “Aardbei”. Gebruik de index waarde om “Aardbei” in jouw console uit te printen.

Opdracht 5:

We gaan het iets complexer maken met het bepalen van de fruit. Er zijn nu 2 arrays gedefinieerd, namelijk een array met getallen en een array met fruitsoorten. De eerste array bepaalt welke fruitsoort er geprint moet worden. Gebruik de Math.random() methode om te bepalen welke waarde uit de eerste array wordt gehaald. Vervolgens kun je de waarde gebruiken als index en de fruitsoort printen in jouw console. Helaas, de getallen 5 t/m 10 uit de eerste array zullen een undefined teruggeven als je dat als index bij de fruit array zal gebruiken. Hiervoor kun je de modulo (%) gebruiken. Zoek uit hoe modulo werkt en pas dat toe aan de waarde van de eerste array. Als resultaat zal er altijd een fruitsoort uitgeprint worden.

Opdracht 6:

Je hebt nu geleerd hoe je arrays kunt aanmaken en de waarde kunt benaderen. Arrays hebben ook functies die je kunt gebruiken. Zoek uit wat push(), pop(), shift(), slice(), splice(), sort(). Maak gebruik van de fruits array en maak voor elke functie een voorbeeld en geef in commentaar wat het doet.

Opdracht 7:

Naast de functies die je van de array kunt gebruiken heb je ook nog de “length” property die je kunt gebruiken. Hiermee kan je de grootte van de array opvragen. Gebruik de length property en print in de console uit hoe groot de fruits array is.

Opdracht 8:

De bestaande values van de array kan je ook aanpassen. Zoek uit hoe je op een specifieke index de waarde kan aanpassen. Pas de value “Peer” naar “Sinaasappel”.

Opdracht 9:

Kijkend naar de fruits array. Hoe draai je de values “Aardbei” en “Mandarijn” om in de array door gebruik te maken van indices (index).

Loops & Iterators

Loops worden gebruikt om op stapsgewijs door de array heen te lopen. Lees hier meer over loops: https://javascript.info/while-for

Leerdoelen:

- For loops
- For in loops
- For of loops
- While loops
- Do while loops

Prerequisites
Maak een nieuwe html bestand aan in VSCode. Copy de volgende code in jouw html bestand.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Loops</title>
</head>
<body>

   <div id="answer"></div>
   <script>
       const sports
 = ["voetbal", "basketbal", "volleybal", "hockey", "honkbal",
"zwemmen", "kortbal"]
   </script>
</body>
</html>
```

Opdracht 1:

Maak gebruik van een loop en itereer de sports array heen. Print het 1 voor 1 uit in jouw console.

Opdracht 2:

Maak een for-loop die alleen even getallen van 0 t/m 20 in je console uitprint.

Opdracht 3:

Maak gebruik van een loop en itereer door de sports array heen. Itereer de loop met even getallen en print de waarde in jouw console uit.

Opdracht 4:

Maak opdracht 1, maar maak nu gebruik van een while loop.

Opdracht 5:

Maak opdracht 1, maar maak nu gebruik van een for...in loop.

Opdracht 6:

Maak opdracht 1, maar maak nu gebruik van een for...of loop.

Opdracht 7:

- Maak een while loop die 3x itereert
- Binnen de while loop maak je een for-loop die 10x itereert
- Bij de eerste while iteratie wil ik dat de getallen 1,2,3,4,5,6,7,8,9,10 in je console uitgeprint wordt
- Bij de tweede while iteratie wil ik dat de getallen 2,4,6,8,10,12,14,16,18,20 in je console uitgeprint wordt
- Bij de derde while iteratie wil ik dat de getallen 3,6,9,12,15,18,21,24,27,30 in je console uitgeprint wordt

Opdracht 8:

Maak een for loop die van 0 t/m 50 itereert. Een hele bekende oefening is de fibonacci reeks. Lees je in hoe fibonacci reeks werkt: https://www.fibonicci.com/nl/rij-van-fibonacci/. Neem zelf eerst de tijd om dit uit te coderen, want op internet staat letterlijk het antwoord. Het gaat hier om het aanleren van bepaalde denkwijze om dit op te lossen. Maak hiervoor een stappenplan en werkt dit uit in code.

Opdracht 9:

Stel je hebt het volgende array met random getallen:

- [2,7,5,10,4,9,3,1,8,6]

Bubble sort is een hele bekende sorterings algoritme. Hiermee kun je de array met random getallen sorteren van klein naar groot. Je mag bij deze oefening geen gebruik maken van de sort() functie. Lees je in hoe Bubble sort werkt: https://en.wikipedia.org/wiki/Bubble_sort en codeer de sorterings algoritme uit.

Objects

Objects zijn essentiële building blocks om data te groeperen. Bijvoorbeeld, je hebt te maken met klanten. Bij ieder klant houd je de naam, achternaam, leeftijd en hobby bij. Een manier van object aanmaken om de klantgegevens bij te houden is:

```
        const Customer = function(name, surname, age, hobby) {
            this.name = name;
            this.surname = surname;
            this.age = age;
            this.hobby = hobby;
        }

        const customer1 = new Customer("Kim Sing", "Cheng", "1337",
"Gaming");
        const customer2 = new Customer("Floris", "Dam", "1337",
"Schaken");
        const customer3 = new Customer("Tess", "Monis", "1337",
"Winkelen");
```

Objecten in javascript kun je op meerdere manieren aanmaken, namelijk:

- Object Constructor
- Literal constructor
- Function Based
- Prototype Based
- Function and Prototype Based
- Singleton Based

Lees je meer in op:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer#:~:text=An%20object%20initializer%20is%20an,data%20types%20or%20other%20objects.

Voorbeelden van de verschillende manieren hoe je objecten creëer:
https://stackoverflow.com/questions/6843951/which-way-is-best-for-creating-an-object-in-javascript-is-var-necessary-befor#:~:text=There%20is%20no%20best%20way,is%20called%20the%20constructor%20function.

Er is helaas geen “beste” manier om objecten te maken. Ieder manier van object aanmaken heeft zijn pros en cons. Wij leggen de focus op Object constructor en Literal constructors.

Prerequisite
Maak een nieuwe html bestand aan in VSCode. Copy de volgende code in jouw html bestand.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>

   <div id="clubs" class="clubs"></div>

   <script>
      // start assignments
   </script>
</body>
</html>
```

Opdracht 1:

Maak een Object die club informatie bijhoudt:

- Naam van de club
- Type sport
- Aantal leden

Opdracht 2:

Maak een Object van contactgegevens en voeg dat ook toe aan de club object:

- Contactgegevens
    - Adres
    - Telefoonnr
    - Contactpersoon

Opdracht 3:

Het is heel gebruikelijk dat dezelfde objecten in een lijst staan. Maak een lijst aan en voeg 4 Club objecten toe aan de lijst. Itereer over de lijst heen en print de verschillende **clubnamen** uit op jouw pagina.

Opdracht 4:

Gegeven deze code:

```
const Person(name, age) {
    this.name = name;
    this.age = age;
}

const persons = [
    new Person("Jan", 24),
    new Person("Klaas", 15),
    new Person("Yanis", 42),
    new Person("Rupel", 55),
    new Person("Hendrik", 32),
    new Person("Mono", 28),
    new Person("West", 23),
]
```

Sorteer persons op leeftijd van klein naar groot.

Opdracht 5:

Maak nu een Person object door gebruik te maken van Object initializers. Wat zou eventueel de voor en nadelen t.o.v Object constructors?

Opdracht 6:

De nieuwe manier van objecten creëren kan je ook gebruik maken van classes. Lees je in over classes:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes#:~:text=Classes%20are%20a%20template%20for,shared%20with%20ES5%20classalike%20semantics.

Zet opdracht 4 om in classes i.p.v object constructors.

Opdracht 7:

Met al de voorgaande onderdelen ben je nu in staat om een simpele spel te maken, Bijvoorbeeld rock, paper scissor. Durf je het aan om dit te bouwen?

Opdracht 8:

JSON is een hele bekende concept binnen developers. Zoek naar een aantal artikelen over JSON. Belangrijk concept om te kennen.