LEERDOELEN

Leerdoelen:

* Ik kan een variabele gebruiken in mijn JavaScript code √

* Ik maak de juiste keuzen tussen consten let.

Let = variabel, waarde kan wijzigen  
const = constant, waarde wijzigd niet

* Ik gebruik camelCaseNotation

jaDatDoeIk

* Ik ken de primitive types van JavaScript, met als belangrijkste:
  + String 'Dit is een String'
  + Boolean true / false
  + Undefined
  + Null Null
  + Number 1, 2, 3
* Ik kan achterhalen wat voor een type een variable is.

In console typeof + naam

* Ik weet wat Dynamically Typed betekent.

Kan je veranderen

Leerdoelen:

* Ik kan uitleggen wat een command line is. **check**
* Ik kan uitleggen wat een GUI is. **check**
* Ik kan uitleggen waarom sommige taken handiger uit te voeren zijn met de command line ten opzichte van een Graphic User Interface (GUI). **check**
* Ik kan een terminal-programma starten en afsluiten **check**
* Ik kan commando's gebruiken om de volgende dingen te doen:
  + bepalen in welke folder ik sta **check**
  + navigeren naar andere folders **check**
  + een nieuwe file aanmaken **check**
  + een file en een folder kopiëren **check**
  + een file en een folder verwijderen **check**
  + zoeken in een grote verzameling bestanden **check**

Leerdoelen:

* Ik kan inline comments schrijven. **Check //**
* Ik kan blok comments schrijven. **Check /\* \*/**
* Ik weet de shortcut in mijn editor om code uit te "commenten". **Shift alt A**

**Leerdoelen:**

* Ik weet wat een comparison operator is.

**<, >, ==, ===, <= , >=, !=**

* Ik kan een == gebruiken

**controleert de waarde**

* Ik kan een === gebruiken   
  **controleert de waarde en het type (string, number etc)**
* Ik kan uitleggen wat het verschil is tussen == en ===

**controleert de waarde en het type (string, number etc)**

* Ik weet wat een logical operator is.

**&& and, || or**

* Ik kan && gebruiken **check**
* Ik kan || gebruiken **check**
* Ik begrijp waarom de titel van deze module geen typefout is.   
  **JS: Comparison && Logical Operators =** JS: Comparison **AND** Logical Operators

**Leerdoelen:**

* Ik kan een object maken in JavaScript **check**

let person = {

name: 'Robin',

age: '12',  
 sport: true

};  
 console.log(person);  
  
 ***=> person is het object, name = property, 'Robin' = value***

* Ik weet hoe ik een property van het object kan aanspreken en gebruiken

1 DOT notation –-- ***person.name = 'Jan'***

2 BRACKET notation –-- ***person['name'] = 'Jan'***

* Ik kan een array maken in JavaScript

**let soortenSport = [**

**'voetbal',**

**'tennis',**

**'hardlopen',**

**'boxen',**

**'icehockey'**

**];**

* Ik kan een element aan een array toevoegen

**let soortenSport [**

**'voetbal',**

**'tennis',**

**'hardlopen',**

**'boxen'**

**];**

**soortenSport ['icehockey']; of**

**soortenSport [14];**

* Ik kan 1 element van de array aanspreken en gebruiken

***console.log(soortenSport[3]); dit is boxen***, want voetbal = 0  
 tennis = 1  
 hardlopen = 2

**Leerdoelen:**

* Ik kan een functie schrijven

|  |  |
| --- | --- |
| function greet () {    console.log('Hello World');  }  greet(); | Function drive () {  console.log('Hello Driver');  } drive(); |
|  |  |
| Function greet (name, lastName) {  console.log('Hello' + ' ' name + ' ' + lastName);  }  greet(Robin, ten Barge); |  |

* Ik kan een functie aanroepen (calling a function)

zie boven  
 greet(); en drive();

Leerdoelen:

* Ik kan functies op een abstract niveau uitschrijven.

**function zetKoffie () {**

**vul koffiezetter met water;  
 doe filter in koffiezetter;**

**doe koffie in filter;**

**}  
 zetKoffie()**

* Ik kan een functie schrijven met parameters.

**function zetKoffie (***aantalSchepjesKoffie***) {**

**vul koffiezetter met water;  
 doe filter in koffiezetter;**

**doe** *aantalSchepjesKoffie***;**

**}  
 zetKoffie(8)**

* Ik kan een functie aanroepen.

z**etKoffie()**

* Ik kan een functie aanroepen met een argument.

**ZetKoffie(8)**

|  |
| --- |
| * Ik kan het resultaat van een functie gebruiken. * Ik weet wat return statements zijn en waarom ik ze wil gebruiken.   **Dit moet ik nog heel goed bekijken:**  [**https://codeburst.io/javascript-what-is-the-return-statement-97d8b11a1a0c**](https://codeburst.io/javascript-what-is-the-return-statement-97d8b11a1a0c) |

Leerdoelen:

* Ik kan het verschil uitleggen tussen const myFunction = function(){} en function myFunction(){}

**Function expression**

-const myFunction = function(){}

Dit is een functie expression, een functie gekoppeld aan een variabele,

bovenstaande is een anonieme function (geen naam)

onderstaande met naam:

const **myFunction** = function **myFunction** (){}

maar kan elke naam bevatten dus kan ook: (beter om bovenstaande te gebruiken)

const **myFunction** = function **zoekFoto** (){}

**Function declaration**

function(){} en function myFunction(){}

Dit is zoals ik het in het begin heb geleerd, recept maken, met broodjeKaas

**Verschil:**

**Bij een function declaration** kan je eerst de code aanroepen opschrijven en dan de  
functiedeclaration zie vb

vb

console.log(doeIets);

doeIets();

function doeIets() {

console.log('declare something');

}

=> wordt de hele functie uitgeprint in de console en de uitvoering van de function

**Bij een function expression**  kan je **NIET** eerst de code aanroepen opschrijven en dan de  
expression zie vb

console.log(doeIets);

doeIets();

let doeIets = function () {

console.log(éxpress something');

};

=> bij de function expression krijgen we als melding dat de variabele is undefined

bij het uitvoeren krijgen we een error

dit omdat de js-engine nog niet bij de regel is aangekomen waar de function wordt gekoppeld aan de variabele,  
en weet dus niet dat de variabele een function is.

En dus de functie niet kan uitvoeren

**==>Je kan dus functions koppelen aan variabelen<==**

Voorbeeld + uitleggen

|  |  |
| --- | --- |
| var runIt = function (a) {  console.log(a);  a();  }; | =>function expression gekoppeld aan variabele  runIt  -je nu iets in in de function zetten (a)  -logt naar console  voert uit a(); |
|  | =>nu kan je dus in function (a) een andere functie zetten |
| runIt(); | Met runIt(); roep je de runIt functie aan  en wordt    var runIt = function (a) {  console.log(a);  a();  };  uitgevoerd |
| runIt (function() {  var b = “Now”;  console.log(“Running” + b );  } ) ; | Nu roepen we de runIt function aan  en geven het een function (zonder naam\*) met de code  { var b = “Now”; console.log(“Running” + b);  }); |
| \*handig om de functie een naam te geven zodat je deze kunt teruglezen in de callstack |  |
|  |  |

**Leerdoelen:**

* ik kan een while loop schrijven

|  |  |
| --- | --- |
| 1 schrijf een variabelen  vb  let age =5 |  |
| while() | Hier staat wanneer deze conditie true is dan |
| while(condition) |  |
| {code} | Voer deze code uit |
|  | Wat er dan gebeurd is, de code wordt 1x uitgevoerd, gaat weer terug naar de top  dus naar while(condition) is deze true dan voert hem voor de 2de keer uit, dan weer terug naar while(condition)...etc etc |
|  | Het is dus belangrijk om te zorgen dat de contitie op een gegeven moment false is anders blijft hij loopen en crashed susteem |
| dus |  |
| Let age =20  while (age < 10 ) {  console.log('Your age is less then 30');  }  console.log ('Your old') | Variabele is 5  wanneer je leeftijd kleiner is dan 10 log op de console 'Your age is less then 10'  deze loop blijft draaien, want age is onder de 30 |
| Dus | Ergens moeten we zorgen dat de age boven de 30 komt zodat de loop onderbroken wordt |
| Let age =20  while (age < 10 ) {  console.log('Your age is less then 30');  age++;  }  console.log ('Your old') | Dit op de volgende manier  age++ dit is de plus 1 operator, dus elke keer dat de loop draait komt er 1 bij, totdat deze 30 is en dan stopt de loop  en  wordt op de console ('Your old') gelogd |

* **ik kan een for loop schrijven**

|  |  |
| --- | --- |
| Dit is hetzelfde als een while loop maar wordt ander geschreven |  |
|  |  |
| While loop |  |
|  |  |
| Let age =20  while (age < 10 ) {  console.log('Your age is less then 30');  age++;  }  console.log ('Your old') |  |
|  |  |
| For loop |  |
| Ook hier heb je de index variabele nodig en dat is  let age = 20 | variabele |
| Dan heb je de conditie nodig en dat is  (age < 10) | voorwaarde |
| Daar na heb je de incrementer nodig en dat is  age++ | Plus 1 incrementer |
|  |  |
| For loop |  |
|  |  |
| For (index variabele; conditie; incrementer) {  console.log('Your age is less then 30');  }  console.log ('Your old') | |
| For (age=20; age<10; a++) {  console.log('Your age is less then 30');  }  console.log ('Your old') | |
| Bij for loop start bij 0 |  |
| Dus bij voorbeeld loop door een site en vindt alle links en voer voor elke link een code uit of style iets | |
|  |  |
| for ( i = 1; i <= links.length; i++) {      console.log('this is link number' + i);  }  console.log ('all links now looped') |  |
|  |  |
|  |  |

**De forEach() methode voert de meegegeven functie 1 keer uit voor elk array element.**

**Iterators**

Het Engelse "to iterate" betekent "iets herhaaldelijk doen". Dingen herhaaldelijk doen is iets dat je in software regelmatig gaat doen. Eén herhaling noemen we dan één "iteratie".