THOMAS MORE

JavaScript

Syntaxt & controlestructuren

Inhoudsopgave

THOMAS MORE

JavaScript

Algemene informatie

Kennismaking

Lesmateriaal

- Canvas
  - PowerPoints
  - Voorbeelden (theoriebestanden)
  - Oefeningen
- Cursuswebsite
  - https://javascript.pit-graduaten.be/
- Bronnen
  - https://developer.mozilla.org
  - https://www.w3schools.com/js/default.asp
  - Google !!!

Voorkennis

- HTML, CSS & Bootstrap

ECMAScript - JavaScript

- JavaScript is gestandaardiseerd door Ecma International
- Ecma International is een Europese vereniging voor het standaardiseren van informatie- en communicatiesystemen
- ECMA was vroeger een acroniem voor **E**uropean **C**omputer **M**anufacturers **A**ssociation
- JavaScript is gedocumenteerd in ECMA-262
- Momenteel => **ECMAScript® 2024** language specification
  - 15th edition (Jun 2024)
  - https://262.ecma-international.org/

JavaScript?

Waar kan ik JavaScript downloaden?

- Eerste lessen:
  - https://bun.sh
  - Installatieinstructies: https://javascript.pit-graduaten.be/lessen/devenv/javascript
- Vanaf hoofdstuk 4:
  - JavaScript wordt uitgevoerd in uw browser op uw computer, op uw tablet en op uw smartphone (client side)
- Vanaf hoofdstuk 6/JS Advanced:
  - https://nodejs.org
  - Installatieinstructies: https://javascript.pit-graduaten.be/lessen/devenv/

JavaScript?

Is JavaScript gratis?

- JavaScript is voor iedereen **gratis** te gebruiken

JavaScript is een van de 3 talen die alle webdevs moeten leren:

- **HTML** om de inhoud van webpagina's te definiëren
- **CSS** om de lay-out van webpagina's te specificeren
- **JavaScript** om het gedrag van webpagina's te programmeren

Waarom JavaScript

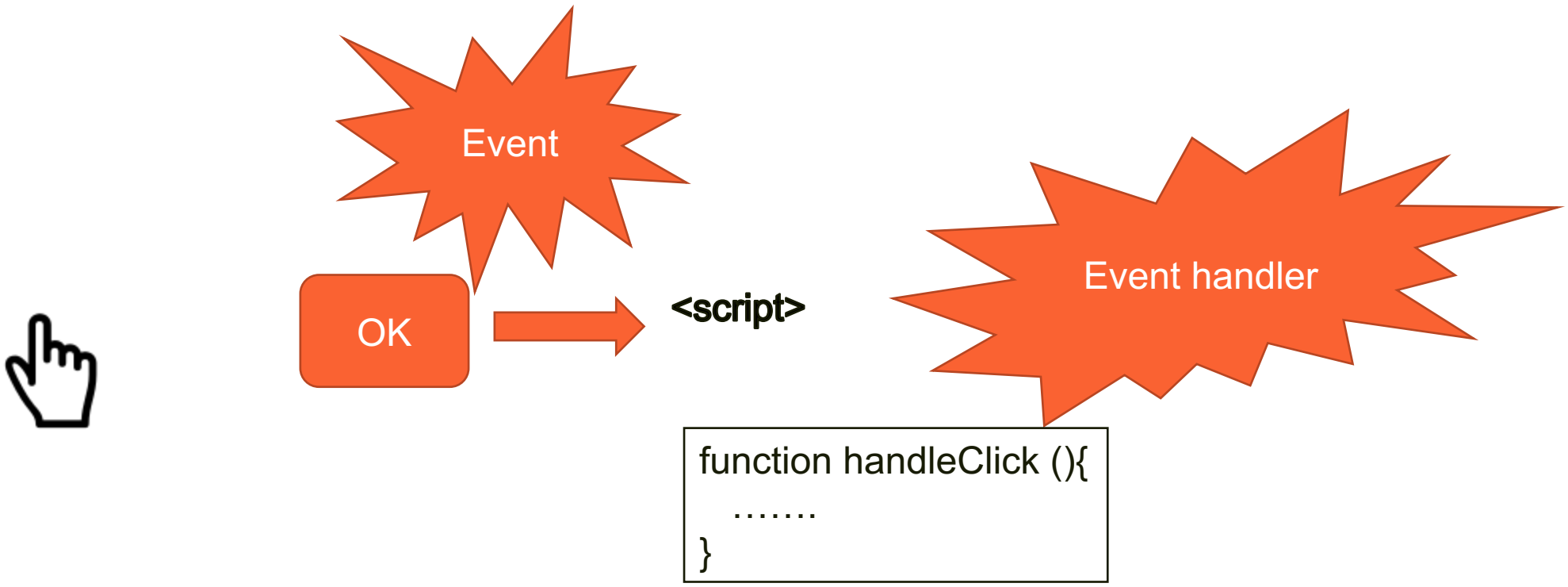
- Elementen toevoegen, wijzigen of verwijderen
- Attributen van elementen wijzigen
- Formuliervalidatie
- Dynamische menu's en afbeeldingen
- Aanpassingen van stijlen en animatie
- Ajax-webapplicaties
- Bootstrap -> nav, modal, carousel, accordion…
- Single page apps waar de UI volledig in JavaScript gebouwd wordt
- Database aanspreken (op een server)
- …

JavaScript

Bestaat uit set van instructies:

- Variabelen
- Lussen
- Teksten
- Arrays
- Objecten
- En nog zoveel meer..

Events en actions (event handlers)



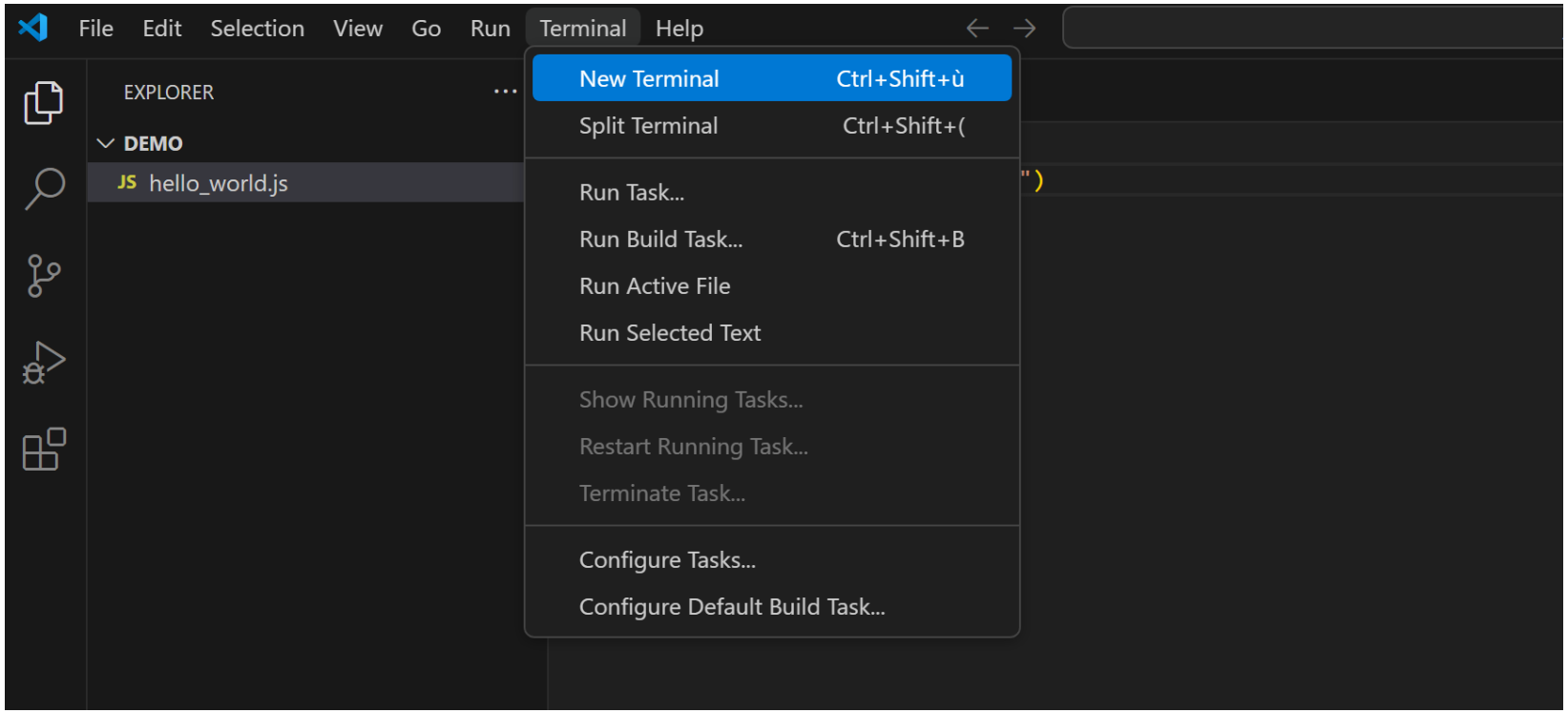
THOMAS MORE

JavaScript

Script uitvoeren

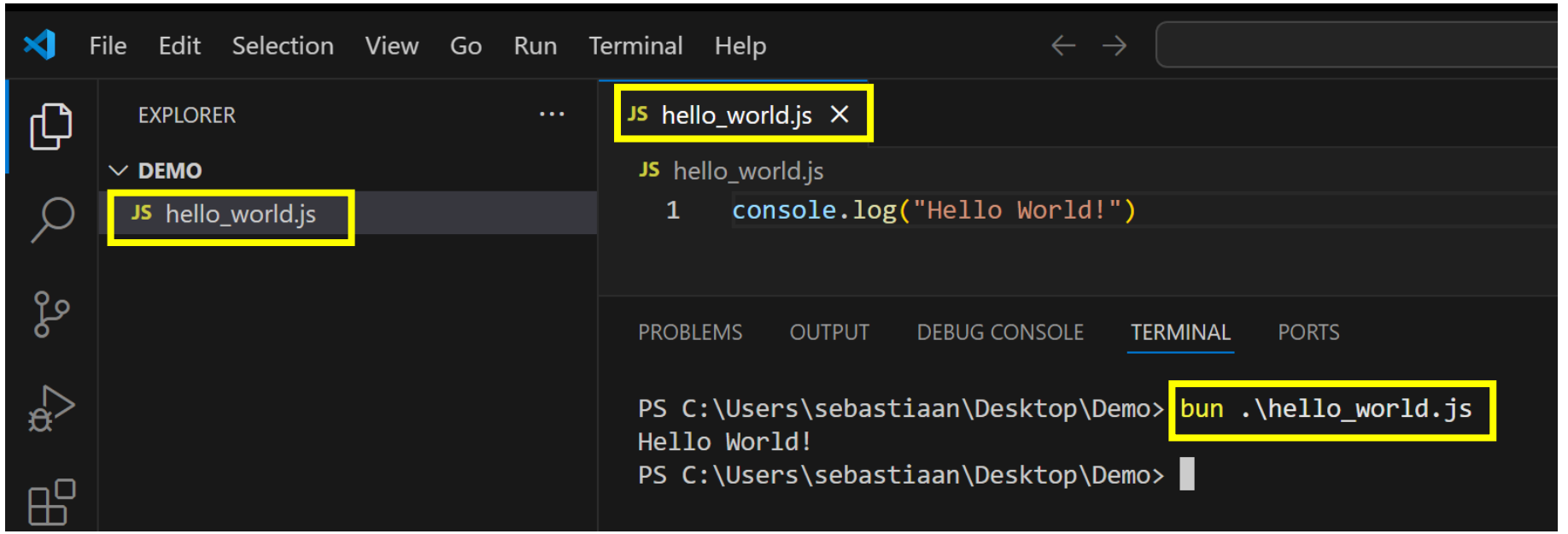
Script uitvoeren

1. Open de map die het script bevat in VS Code
2. Open de terminal is VS Code



Script uitvoeren

1. Open de map die het script bevat in VS Code
2. Open de terminal is VS Code
3. Type *bun $SCRIPT_NAAM* in de terminal en druk op enter



Script uitvoeren

1. Open de map die het script bevat in VS Code
2. Open de terminal is VS Code
3. Type *bun $SCRIPT_NAAM* in de terminal en druk op enter
   - Tab vult automatisch aan: *bun he* + *tab* → *bun hello_world.js*
   - Pijltoetsen navigeren door de command geschiedenis

THOMAS MORE

JavaScript

Syntax regels

Syntax regels (1)

- Afzonderlijke woorden scheiden door één of meerdere spaties.
- Statement eindigt steeds met een puntkomma, maar dit is niet verplicht.
- Variabelen/constanten (let, const)
  - beginnen steeds met een **kleine** letter, een underscore _ of een dollarteken $, maar mogen nooit beginnen met een cijfer.

Syntax regels (2)

- **Strings** kunnen zowel tussen **enkele** als tussen **dubbele aanhalingstekens** staan
  - Enkele aanhalingstekens binnen dubbele aanhalingstekens (of omgekeerd) zijn eveneens toegestaan.
- Gebruik een **backslash** als **escape karakter**.
  - Het karakter erna wordt dan speciaal geïnterpreteerd
- JavaScript is **hoofdlettergevoelig**
  - Zowel variabelen, functies als objecten zijn hoofdlettergevoelig
- Voeg **commentaar** toe aan je script
  - Commentaar op één lijn wordt voorafgegaan door //
  - Commentaar over meerdere lijnen plaatst u tussen /* en */

THOMAS MORE

JavaScript

Variabelen & datatypes

Variabelen

- Kleine **containers** om **informatie** in te **bewaren**
- Voorbeelden

```
let a = 7;          // a is een getal
a = "welkom";       // a is vanaf nu een string
const loop = 5;
const admin = "Niels";
```

STRINGS

02_variabelen.js

- Soorten
  - Getallen - Strings
  - Booleaanse waarden - Objecten - Arrays
  - null - undefined

- Datatype controleren
  - typeof variable

Variabelen

https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/

- **let** variabelen kunnen aangepast worden, maar **niet** geredeclareerd
  - Gebruiken wanneer een variabele later misschien een andere waarde kan krijgen.
- **const** variabelen kunnen **niet** aangepast of geredeclareerd worden
  - Gebruiken wanneer een variabele niet meer wijzigt van waarde.
  - Altijd de **eerste** keuze
- **var** variabelen kunnen **aangepast** en **geredeclareerd** worden
  - Absoluut NIET gebruiken. Waarde variabele kan overal aangepast worden.

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

STRINGS

03_strings.js

- Enkele of dubbele quotes

```
const sgl = 'Single quotes. ';
const dbl = "Double quotes";
```

- Zelfde quotes ‘escapen’

```
const bigmouth = 'I\'ve got no right to take my place...';
```

- Samenvoegen strings

```
const one = 'Hello, ‘;
const two = 'how are you?’;
const response = one + 'I am fine – ' + two;
```

- Template literals (gebruik backticks `` !)

```
const response = `${one} I am fine – ${two}`;
```

operatoren

- Rekenkundige operatoren
  - * / + - % ++ -- ** += -=
- Vergelijkende operatoren
  - == === (zelfde datatype, de **standaard** keuze)
    < >
    <= >=
    != !== (zelfde datatype, de **standaard** keuze)
- Logische operatoren
  - ! (not) && (and) || (or)
- String operatoren
  - + +=

JavaScript output

- Schrijf naar de **browser/terminal console**

```
console.log("string")
console.log("string", "string2")
console.log("string", 1, "string2")
console.log("string", variabele, "string2")
```

prompt()

04_prompt.js

- Vraag **invoer** aan de gebruiker
  - Bun → console input
  - Browser → popup venster met inputveld

```
// Vraag invoer aan de gebruiker, de tweede (optionele) parameter is de default waarde.
const postcode = prompt('Vul uw postcode in:', '2440');
const tekst = `De postcode die u invoerde was: ${postcode}`;
console.log(tekst);
```

prompt()

05_prompt2.js

- Vraag **invoer** aan de gebruiker
  - Bun → console input
  - Browser → popup venster met inputveld
  - Invoer is altijd een **string**
    - Converteren via de *Number()* constructor

```
const age = Number(prompt('Hoe oud ben je'));
const text = `Je bent minstens ${age * 365} dagen oud.`;
console.log(text);
```

confirm()

06_confirm.js

- Vraag **bevestiging** aan de gebruiker
  - Bun → console input
  - Browser → confirmatie box (OK/Cancel)
  - Geeft een *boolean* terug

```
// Vraag invoer aan de gebruiker, de tweede (optionele) parameter is de default waarde.
const postcode = prompt('Vul uw postcode in:', '2440');

// Vraag bevestiging aan de gebruiker, default is false (nee).
const success = confirm(`Is ${postcode} de correcte postcode?`);
console.log('Success:', success)
```

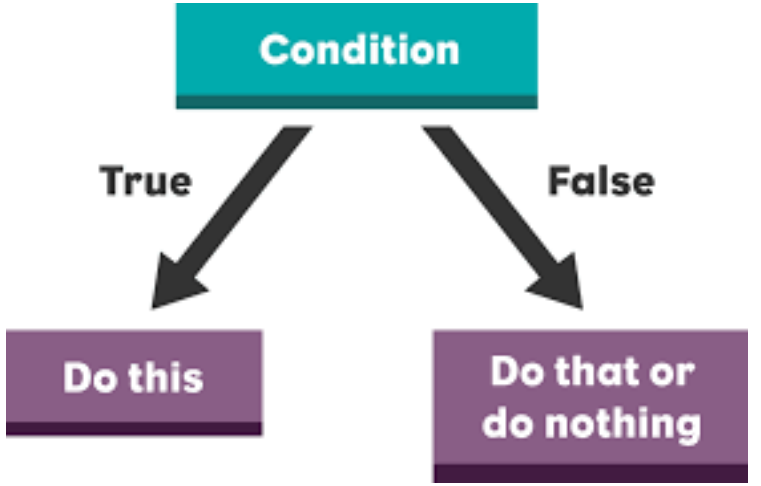
THOMAS MORE

JavaScript

Voorwaardelijke instructies

If - else

- Voorwaarde kan 2 waarden hebben
  - Waar (true / 1)
    - voer dan alle instructies uit binnen het **if-blok**.
  - Onwaar (false / 0)
    - voer dan de instructies binnen het **else-blok** uit.



If – else – else if syntax

// if met **één voorwaarde**
```
if (voorwaarde) {
    // voer instructie(s) uit indien voorwaarde waar is
}
```

// if met **één voorwaarde**
```
if (time < 18) {
    greeting = "Good morning";
}
```

// if-else met **één voorwaarde**
```
if (voorwaarde) {
    // voer instructie(s) uit indien voorwaarde waar is
} else {
    // anders voer instructie2 uit
}
```

// if-else met **één voorwaarde**
```
if (time < 18) {
    greeting = "Good morning";
} else {
    greeting = "Good evening";
}
```

// if-else met **meerdere voorwaarden**
```
if (voorwaarde1) {
    // voer instructie(s) uit indien voorwaarde1 waar is.
} else if (voorwaarde2) {
    // voer instructie(s) uit indien voorwaarde2 waar is.
} else {
    // anders voer deze instructie(s) uit
}
```

// if-else met **meerdere voorwaarden**
```
if (time < 10) {
    greeting = "Good morning";
} else if (time < 20) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

Meervoudige keuze

07_if_else_if_else.js

- **Simpele** vorm
  - `if … else`
- **Geneste** vorm
  - `if … else if … else`
  - `if … else if … else if … else`
- **Zonder accolades** kan, indien maar **1 statement** moet uitgevoerd worden
  - Met of zonder newline

```
if(name === myName && age === myAge) alert("Ben jij mijn dubbelganger?");

if(name === myName && age === myAge)
  alert("Ben jij mijn dubbelganger?");
```

Ternary expression

08_ternary.js

(voorwaarde) **?** instructie indien waar **:** instructie indien niet waar **;**

```
const dag = prompt("Geef de dag in");
const uitspraak = (dag === "zaterdag" || dag === "zondag") ? "Het is weekend" : "Helaas, nog even werken";
console.log(uitspraak);
```

Voorbeeld

09_voorbeeld.js

```
// 1. Initialisaties
const question = 'Hoeveel is 20 + 60 ?';
const answer = 80;
const correct = 'Prima, ga zo door!';
const wrong  = 'Jammer, volgende keer beter!';

// 2. Vraag stellen
const response = Number(prompt(question, '0'));

// 3. Controleer het antwoord
const result = (response === answer) ? correct : wrong;

// 4. Schrijf resultaat naar de terminal
console.log(result);
```

Switch

10_switch.html

Test de waarde van een variabele in functie van een aantal voorgedefinieerde waardes.

```
switch (meestal een variabele) {
        case voorwaarde1:
             programmacode;
             break;
        case voorwaarde2:
             programmacode;
             break;
        default:
             programmacode;
}
```

- programmacode wordt uitgevoerd als testvoorwaarde gelijk is aan bijhorende case-uitdrukking
- **break**: de code stopt met verdere uitvoer van de programmacode in het switch-statement. Als **break** niet aanwezig is, worden alle andere cases ook doorlopen!
- **default**: wordt uitgevoerd als geen enkele uitdrukking voldoet
- switch statement begint en eindigt met “{“ en “}”

KEUZE IF / SWITCH

11_if_vs_switch.html

Wanneer kies je voor de **IF**

- expressies testen op **waar**
  - Op **bereik** (10<i && i <20)
  - **Verschillende** voorwaarden
- **Verschillende** uitkomsten

- Minder overzichtelijk

Wanneer kies je voor de **SWITCH**

- expressie vergelijken met een **integer / string / (enum)**

- Wanneer bepaalde waarden **dezelfde** uitvoering vereisen

- Overzichtelijker

THOMAS MORE

JavaScript

Lussen

Overzicht iteraties

- **Begrensde** herhaling
  - Geef **10** keer een getal in
- **Voorwaardelijke** herhaling
  - Met aanvangsvoorwaarde
  - Met afbreekvoorwaarde

  Voorbeelden:
  - Een teller verhogen **zolang** deze kleiner of gelijk is dan x
  - Een getal raden **totdat** het (gevonden is) gelijk is aan x

For syntax

12_for.js

- Begrensde herhaling

```
for ( startwaarde ; eindwaarde ; verhoging ) {
        // herhaalt n-maal dezelfde instructie.
}
```

```
for (let i = 0; i < 3; i++) {
  console.log(`Lus ${i + 1}`) ;
}
```

```
// Ook negatieve stapwaarde!
for (let i = 10; i >= 0; i--) {
  console.log(`Lus ${i + 1}`);
}
```

While syntax

13_while.js

- Voorwaardelijke herhaling met **aanvangsvoorwaarde**

```
while(voorwaarde){
        programmacode;
}
```

- zolang de voorwaarde **waar** (true) is gaan we door de lus
- lus stopt wanneer de voorwaarde **onwaar** (false) wordt
- wordt **0 of meerdere keren** uitgevoerd

While syntax

13_while.js

```
let i = 0;

while (i < 10) {
  console.log(`Lus ${i + 1}`);
  i++;
}
```

Do while

14_do_while.js

- Voorwaardelijke herhaling met **afbreekvoorwaarde**

```
do {
        programmacode;
} while (voorwaarde)
```

- zolang de voorwaarde **waar** (true) is gaan we door de lus
- lus stopt wanneer de voorwaarde **onwaar** (false) wordt
- wordt **minstens 1 maal** uitgevoerd

Do while

14_do_while.js

```
let i = 0

do {
  console.log(i)
  i++
} while (i < 10)
```

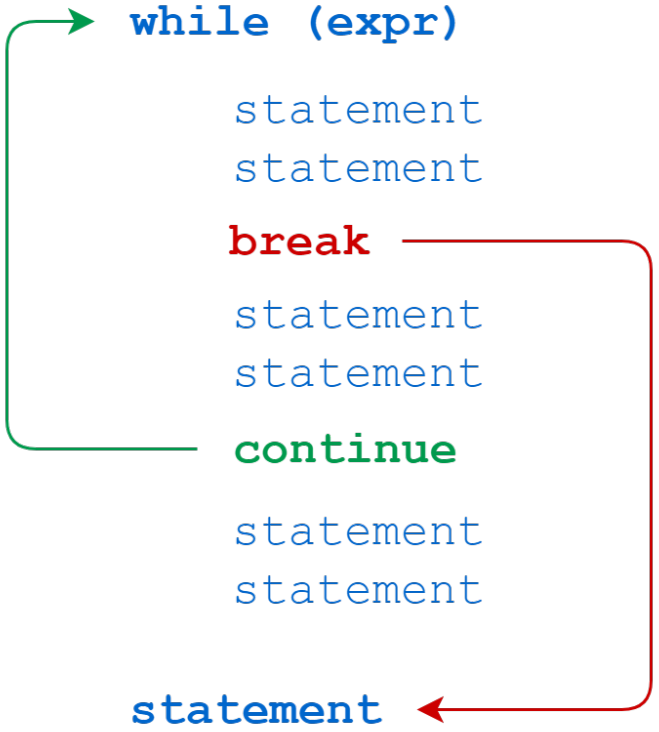
```
Uitvoer
--------

0
1
2
3
4
5
6
7
8
9
```

```
let i = 0

do {
  i++
  console.log(i)
} while (i < 10)
```

```
Uitvoer
--------

1
2
3
4
5
6
7
8
9
10
```

Break vs Continue

15_break_vs_continue.js



- Met **break** en **continue** kun je makkelijk de lus beïnvloeden
- **Break** om de herhaling af te breken
  - Code in de lus NA de break wordt niet meer uitgevoerd
  - Lus stopt onmiddellijk na de break
- **Continue** om statements in huidige lus over te slaan
  - Code in de lus NA de continue wordt niet meer uitgevoerd
  - Lus gaat verder naar de volgende herhaling

- De **break** en **continue** worden vaak in combinatie met de voorwaardelijke struktuur gebruikt.

Valkuil bij continue

16_continue_valkuil.js

```
// Oneindige lus!
let getal = 10,teller = 1;

while (teller <= 10) {
  // teller++; // Oplossing.

  console.log(`${teller} maal ${getal} = ${teller * getal}`);
  if (teller % 2 !== 0) {
    continue;
  }

  // Teller zal nooit verhoogd worden...
  teller++;
}
```

While vs FOR vs DO While

17_for_vs_while.js

Hetzelfde resultaat bereiken

```
for (i=1;i<=10;i++) {
   …
}
```

```
i= 1;
while (i <= 10) {
   …
   i++;
}
```

```
i = 1;
do {
   …
   i++;
} while (i <= 10);
```

THOMAS MORE

JavaScript

Functies

Functie: definitie & doel

- **Definitie**

  Een functie is een codefragment of een variabele die naar de functie verwijst dat door andere code of door zichzelf kan worden aangeroepen.

  Wanneer een functie wordt aangeroepen, worden argumenten als invoer aan de functie doorgegeven en kan de functie optioneel een waarde retourneren.

  Een functie in JavaScript is een object.

- **Doel**
  - **verhogen** het overzicht
  - **hergebruik** van stukken code
  - gebruik door **anderen**

Functie declaratie

18_functies.js

- Zonder argumenten (parameters)

```
function toonBoodschap() {
  console.log("Hallo iedereen!");
}
```

- Met argumenten, eventueel *optioneel*

```
function toonBoodschapDef(naam, voornaam="Thomas") {
   console.log("Hallo" + naam + " " + voornaam);
}
```

Functie aanroepen

19_functie_return.js

```
// gewone functieaanroep: eindig met ;
functieNaam();
```

bijvoorbeeld => startSpel();

```
functieNaam(parameter1, parameter2,...);
```

bijvoorbeeld => startSpel(levens, level, …);

```
// verder werken met de geretourneerde waarde
const mijnVar = functieNaam(param1, param2,…);
```

bijvoorbeeld => let opp = berekenOpp(b,h);

HOISTING

20_hoisting.js

- Declaraties van functies worden in het **geheugen opgeslagen** tijdens de **compilatiefase**, maar blijven precies staan waar je ze in je code hebt getypt.

```
test(); // Dit werkt omwille van hoisting!

function test() {
    alert("Ik ben een functie");
}

test();
```

Functies als variabelen

21_functies_als_variabelen.js

Elke functie kan gebruikt worden als **variabele**

- De ronde haken moet weggelaten worden

```
function executeNTimes(n, callback) {
    for (let i = 0; i < n; i++) {
        callback();
    }
}

function printHello() {
    console.log("Hello");
}

executeNTimes(5, printHello);
```

Functies als variabelen

21_functies_als_variabelen.js

Anonieme functie heeft geen naam

- Moet gekoppeld worden aan variabelen

```
function executeNTimes(n, callback) {
    for (let i = 0; i < n; i++) {
        callback();
    }
}

executeNTimes(5, function() {
    console.log("Anonieme functie");
})
```

Scope van variabelen

22_functie_scope1.js

- Bij functie aanroep
  - wijzigingen van **variabelen** die **in** de **functie gedefinieerd** worden hebben geen invloed op de waarde van **gelijknamige variabelen buiten** de **functie**.

- Principe “call by value”
  - wijziging van argument geen invloed op variabele buiten de functie

Scope van variabelen

22_functie_scope2.js

- De functie heeft volledige toegang tot de **variabele buiten** de functie. Deze kunnen ook **wijzigen** binnen de functie.

```
let username = "John";

console.log(username); // John voor de functieaanroep
toonBoodschap();
console.log(username); // Bob, de functie heeft de waarde aangepast

function toonBoodschap() {
      username = "Bob"; // veranderd de buiten variabele
      const boodschap = "Hello, " + username;
      console.log(boodschap); // Hello Bob
}
```

Scope van variabelen

- Een variabele (var/let) kan **overschreven** worden **binnen** een functie, waarde van variabele **buiten** de functie blijft **ongewijzigd**.

```
let username = "John";

console.log(username); // John
toonBoodschap();
console.log(username); // John, onveranderd, neemt userName buiten de functie

function toonBoodschap() {
      // Declareert een nieuwe variabele binnen de functie
      const username = "Bob";
      const boodschap = "Hello, " + userName;
      console.log(boodschap); // Hello Bob
}
```

Oefeningen