Hoofdstuk 2: Controlestructuren

Doel van controlestructuren

De uitvoering van een programma verloopt nooit in een rechte lijn van start naar einde. Het programma krijgt input van de gebruiker of van andere programma's en dient daar gepast op te reageren door de juiste code uit te voeren. Dit wordt gedaan via **controlestructuren** die de **flow** van het **programma controleren**.

Relationele operatoren

Om het verloop van een programma te kunnen controleren wordt er gebruik gemaakt van relationele operatoren. Met deze operatoren kan je **twee waarden met elkaar vergelijken**. Het resultaat van deze vergelijking is altijd een **boolean** (true of false). Je kan de volgende operatoren gebruiken in JavaScript:

- a **>** b: is a groter dan b?
- a **>=** b: is a groter of gelijk aan b?
- a **<** b: is a kleiner dan b?
- a **<=** b: is a kleiner of gelijk aan b?
- a **==** b: is a gelijk aan b?
- a **===** b: is a gelijk aan b en hebben ze ook hetzelfde datatype?
- a **!=** b: is a verschillend van b zonder te letten op het datatype?
- a **!==** b: is a verschillend van b, ook gelet op het datatype?

Je kan een demo van de operatoren vinden op https://codepen.io/fdc/pen/RwbVrKr. De meeste zijn redelijk voor de hand liggend. Merk wel dat er een verschil is tussen == en ===. Als je drie = tekens gebruikt wil je dat a en b ook van hetzelfde type zijn (meestal allebei number of allebei string). Standaard gebruiken we binnen deze cursus de 3 gelijkheidstekens.

Booleaanse operatoren

OPMERKING: Je zal in plaats van booleaanse operatoren ook af en toe de naam logische operatoren horen.

Je weet intussen dat een relationele operator een boolean (true of false) als resultaat geeft. Met een booleaanse operator kan je **twee of meerdere booleans ten opzichte van elkaar evalueren**. Het resultaat is opnieuw een boolean. Je kan in JavaScript drie booleaanse operatoren gebruiken.

- **AND** operator (&&)
  - Het resultaat is enkel true als **alle condities** true zijn
- **OR** operator (||)
  - Het resultaat is true als **één** of meerdere **condities** true zijn
  - de rechte streep is het pipe character. Op je Mac tik je dit via SHIFT + OPT + L
- **NOT** operator (!)
  - Het resultaat is enkel true als de input false is. De operator wordt vooral gebruikt om een waarde **om te keren**

Onderstaande tabel geeft een overzicht van de verschillende operatoren en hun uitkomst. Je kan ook een demo bekijken op https://codepen.io/fdc/pen/ExYmPdq.

Input 1	Input 2	AND	OR	NOT (Input 1)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Booleaans operatoren zijn heel eenvoudig in gebruik zolang het maar over twee inputs gaat. Je kan ook complexere constructies opzetten met meerdere vergelijkingen. In dit geval krijgt de AND operator voorrang op de OR operator. Je gebruikt **ronde haakjes** om deze voorrang op te heffen, net zoals je bij wiskunde haakjes gebruikt om voorrang te geven. Een voorbeeld kan je vinden op https://codepen.io/fdc/pen/BaBRKJq.

TIP: het is aan te raden om altijd haakjes te gebruiken bij meerdere vergelijkingen. Dit zorgt er voor dat je code beter leesbaar is.

Controlestructuren in JavaScript

Booleaanse en relationele operatoren hebben op zich weinig meerwaarde. Je krijgt enkel true of false als antwoord. Ze worden pas krachtig als ze gebruikt worden in een controlestructuur. Deze controlestructuur zal afhankelijk van true of false **bepalen hoe het programma verder moet lopen**.

if

De if structuur zal een stuk code enkel uitvoeren als er aan een conditie voldaan is. Je kan deze structuur dan ook letterlijk lezen als "**als** er aan de **voorwaarde voldaan** is **dan voer** je de volgende actie(s) **uit**". Bijvoorbeeld: "als de leeftijd van de gebruiker groter of gelijk is aan 18 dan mag de gebruiker de website bezoeken".
De syntax van een if structuur ziet er als volgt uit. Je gebruik het keyword if, gevolgd door de conditie (voorwaarde) die je wil testen tussen haakjes. De blok code die moet uitgevoerd worden indien de conditie waar is staat tussen accolades.

```
if(voorwaarde === true){
   console.log('de voorwaarde is waar');
}
```

De if structuur beoordeelt de conditie telkens op basis van een boolean

- Is de input van de conditie een string of number volgt er een relationele operator om tot een boolean te komen.
- Is de input van de conditie een boolean op zich mag je de relationele operator weglaten.

Een demo kan je vinden op https://codepen.io/fdc/pen/QWLvEww

if … else

De if … else structuur is een uitbreiding op de if structuur. Waar je bij de if structuur enkel aangeeft wat er moet gebeuren indien de conditie waar is kan je bij de if … else structuur ook aangeven wat er moet gebeuren indien de conditie niet waar is. Je leest dit als "**als** er aan de **voorwaarde voldaan** is **dan voer** je de volgende actie(s) **uit**, **anders** voer je een **andere actie** uit". Bijvoorbeeld: "als de leeftijd van de gebruiker groter of gelijk is aan 18 dan mag de gebruiker dan website bezoeken, anders ga je naar de site van studio 100".
De structuur is gelijkaardig aan een if structuur. Je hebt nog steeds een conditie die resulteert in true of false. Maar je geeft ook aan wat er moet gebeuren indien de conditie false is.

```
if (voorwaarde === true) {
 console.log('de voorwaarde is waar');
}else{
 console.log('de voorwaarde is niet waar')
}
```

Een demo kan je vinden op https://codepen.io/fdc/pen/pozPqRd

OPMERKING: je kan gebruik maken van de ternary operator. Dit is een kortere manier om een if … else structuur te schrijven. De syntax is als volgt: voorwaarde ? indien waar : indien niet waar. Gebruik dit echter met mate. Kortere code is niet altijd leesbare code.

if … else if … else if … else if … else

Soms heb je nood aan een meer complexe structuur. Je kan binnen een else statement opnieuw een if statement oproepen. Je bekomt dan een **geneste if … else structuur**. Een demo kan je vinden op https://codepen.io/fdc/pen/LYPyMLK. Bekijk aandachtig de code en lees ook de comment bovenaan zodat je de redenering goed snapt.

Let bij dergelijke constructies heel goed op de uitlijning van je **code** zodat deze **leesbaar** blijft.

switch … case

Een switch … case structuur is een set van **vooraf gedefinieerde keuzeopties** die de flow van je programma bepalen afhankelijk van de input. Er wordt **exact 1 waarde getoetst** aan verschillende keuzeopties.

De structuur ziet er als volgt uit. bij het **switch** keyword zet je de te checken waarde tussen ronde haakjes. Daarna heb je verschillende **cases** of mogelijkheden. Merk op dat je hier geen range kan ingeven, het moet altijd een **exacte waarde** zijn. Elke case wordt afgesloten met het keyword **break** om te voorkomen dat de overige cases ook uitgevoerd worden.

Indien er geen enkele case overeenkomt met de te checken waarden wordt de **default** case uitgevoerd. Je bent echter niet verplicht om een default op te geven.

```
switch(waarde){
 case `optie 1`:
   // code uitvoeren als waarde = optie 1
   break;
 case `optie 2`:
   // code uitvoeren als waarde = optie 2
   break;
 case `optie 3`:
   // code uitvoeren als waarde = optie 3
   break;
 default:
   // code uitvoeren indien waarde niets van bovenstaande is
}
```

Een demo kan je vinden op https://codepen.io/fdc/pen/VwZbNqv. Verwijder daar eens het break keyword en merk dat alle cases uitgevoerd worden. Pas ook de waarde van richting aan zodat een andere case wordt uitgevoerd.

if of switch?

Meestal kan je een switch … case ook als meerdere if … else statements schrijven. In sommige gevallen is het ook niet direct duidelijk welke controlestructuur je het best gebruikt. De algemene regel is:

- switch: testen op 1 exacte waarde met meerdere cases
- if: testen op een vergelijking met relationele operator

Daarnaast is ook de context en hoe leesbaar de code is een belangrijke parameter om te bepalen welke controlestructuur je het best gebruikt.

Oefeningen 2: Controlestructuren

Maar eerst nog even dit: BYOPS - Bring your own Paper & Stylo

Maak een diagram van volgende stellingen

1. Ben ik geslaagd (score op 20) voor Development 1, dan volgen er felicitaties.
2. Als het 1 juli is, dan moet mijn automatische out of office mail aan staan.
3. Ik kan inloggen als mijn gebruikersnaam én paswoord correct zijn.
4. Is mijn BMI hoger dan 25 moet ik sporten, is het lager dan 20 dan moet ik meer eten. Een BMI van 20 tot 25 wijst me op een gezonde levensstijl.
5. In het studentenrestaurant is er een vast weekmenu; is het maandag, dan eten we vlees, dinsdag = vegetarisch, woensdag = koude schotel, donderdag = frietjes en vrijdag is visdag. wat eten we vandaag?

Oefening 1: IF aanwezig

Kopieer volgende code in een nieuw Javascript document en kijk wat er in je console verschijnt. Verander de boolean naar false en bekijk het opnieuw.

```
let aanwezig = true;
if(aanwezig == true){
  console.log(`persoon x is aanwezig`);
}
```

verwijder "== true" en kijk naar het resultaat. Wat is er veranderd?

Oefening 2: IF… ELSE nieuwsbrief

Kopieer volgende code in een nieuw Javascript document en kijk wat er in je console verschijnt. Verander de boolean naar false en bekijk het opnieuw.

```
let nieuwsbrief = true;

if(nieuwsbrief) {
  console.log('U bent geabonneerd op de nieuwsbrief');
}else{
  console.log('U bent niet ingeschreven op de nieuwsbrief');
}
```

Oefening 3: IF… ELSE Sold Out

Maak een boolean `soldOut`. Toon in de console of er nog tickets zijn voor "Black Box Revelation" in De Kreun / Wilde Westen.
https://www.wildewesten.be/nl/event/black-box-revelation

Oefening 4: IF… ELSE Brusselse Ring

Verkeersdrukte op de Brusselse ring wordt uitgedrukt met een indexschaal van 0 tot 10. Afhankelijk van de index geven de snelheidsborden boven de snelweg volgende waarden aan:
1-5: 120km/u
5-8: 110km/u
8>: 90km/u
Toon dit in je console en test dit door de index aan te passen.

Oefening 5: SWITCH keyboardcontroller

Kopieer volgende code in een nieuw Javascript document en kijk wat er in je console verschijnt. Verander de variabele naar “links” en bekijk het opnieuw.

```
let richting = `vooruit`;
switch (richting) {
case `vooruit`:
      console.log(`Zet een stap vooruit`);
      break;
case `achteruit`:
      console.log(`Zet een stap achteruit`);
      break;
case `links`:
      console.log(`zet een stap naar links`);
      break;
case `rechts`:
      console.log(`Zet een stap naar rechts`);
      break;
}
```

Oefening 6: SWITCH Bananastatus

Maak een string `bananaStatusColor`. Als de banaan “groen” is toon je de boodschap: “nog niet rijp”. “Geel” wil zeggen: “eetklaar!”. “Bruin” geeft aan: “te laat”.

Oefening 7: SWITCH You’re not my type (2)

Maak een variabele `whatsMyType` en ken er een willekeurige waarde aan toe. Toon vervolgens in je console of dit een Boolean, String of Number is, test dit met verschillende variabelen.

Oefening 8: CYA

1. Maak een variabele `userName` en ken er een lege String aan toe.
   Als je gebruiker dit wil, kan hij hier zijn naam invullen.

Als de gebruiker zijn naam heeft ingevuld, toon je op het scherm “Hallo, gebruikersnaam!”. Anders komt er op het scherm “Hallo!”.

2. Maak een variabele “`userQuestion`”. Hieraan ken je een Ja/Nee vraag toe. bvb. “kunnen flamingo’s vliegen?”.
   Toon de vraag op het scherm volgens dit format: “*Noor vraagt: kunnen Flamingo’s vliegen?*”.
3. Maak variabele `randomNumber` en ken er volgende code aan toe: (Deze code genereert een willekeurig geheel getal van 0 tot 8.) `Math.floor(Math.random()*8);`
4. Maak een variabele `eightBall`.
   In deze variabele moet nu, afhankelijk van de randomNumber een andere tekst opgeslagen worden. Voorzie volgende opties:
   1 - Het is zeker
   2 - Het is beslist zo
   3 - Zonder twijfel
   4 - Zeer zeker
   5 - Je kunt erop vertrouwen
   6 - Volgens mij wel
   7 - Zeer waarschijnlijk
   8 - Ja

Toon het resultaat op het scherm via console.log en test dit een paar keer uit.