

JavaScript 2

JavaScript lesson voor eerstejaars AO leerlingen

- [Lesplan](#)
- [Les 1: Herhaling JavaScript I](#)
- [Les 2: For-lussen \(For-loops\)](#)
- [LEGENDA](#)
- [Week 1: Uitwerkingen les - en huiswerk opdrachten](#)
- [Week 2: Uitwerkingen les - en huiswerk opdrachten](#)
- [Les 3: For-loop, Continue - en Break statement](#)
- [Week 3: Uitwerkingen les - /huiswerk opdrachten](#)
- [Week 5: JavaScript functions huiswerk](#)

Lesplan

In de JavaScript lessen zullen we gebruik maken van de stof die is opgedaan in de vorige JavaScript lessen. Deze kennis zullen we vergroten door een stuk van de basis te herhalen, uit te breiden en nieuwe kennis op te doen.

De eerste week van week 1 begint met een instaptoets **[1]**. Deze toets telt niet mee in je eindcijfer en dient als hulpmiddel voor de docent. Na de instaptoets zullen we een beginnen met een stukje JavaScript kennis die is opgedaan in de vorige lessen en verder werken aan het leren van JavaScript skills!

Het kan zijn dat bepaalde begrippen worden uitgelegd aan het einde van de pagina. Deze lijst bevat begrippen die je kunnen helpen bij het begrijpen van de lesstof.

Zie je toch een onbekend begrip? Geef het aan bij de docent!

LET OP!! Neem de LEGENDA door om de pagina's optimaal te gebruiken.

Setup

Operating system:

Windows is de default operating system (OS) die in de lessen gebruikt zal worden. Andere operating systems zijn toegestaan, maar het kan zijn dat de docent niet altijd kan helpen met het troubleshooten van OS gerelateerde problemen.

Webbrowser:

Verder maken we gebruik van de Google Chrome webbrowser:

<https://www.google.com/chrome/index.html> .

Editor:

Let op! Als student kun je met je schoolmail een gratis licentie aanvragen voor WebStorm.

Navigeer naar <https://www.jetbrains.com/shop/eform/students> en vraag een gratis studenten licentie aan!

De Integrated Development Environment (IDE) **[B1]** die we gaan gebruiken is WebStorm. Deze kan via de volgende link gedownload worden: <https://www.jetbrains.com/webstorm/> . Je studentenlicentie kan gebruikt worden om WebStorm te activeren!

Les(opdrachten) en huiswerk

Vanaf week twee begint elke les met het bespreken van het huiswerk. De leerlingen krijgen de ruimte om aan te geven waar ze tegenaan zijn gelopen en wat ze lastig vonden. Hierbij wordt gekeken naar de stappen die de leerling zelf heeft genomen om de problemen op te lossen.

Na de huiswerkbepreking wordt een nieuw stukje theorie behandeld en krijgen de leerlingen de gelegenheid om aan de les-opdrachten - **[2]** en huiswerk te werken.

Een overzicht van de lesstof is te vinden in de studiewijzer hieronder. Voor meer informatie kun je de individuele lessen raadplegen.

Studiewijzer

Week	Verwachting	Beschrijving
------	-------------	--------------

1

Van de leerling wordt verwacht dat hij/zij bekend is met de stof van JavaScript I.

- Instaptoets
- Herhaling JavaScript I:
 - Console
 - Variabelen
 - Operatoren en syntax:
 - Vergelijkingsoperatoren
 - Rekenkundige operatoren
 - Logische operatoren
 - Datatypen:
 - Integer, number, float
 - String
 - Boolean
 - Build-in function: `typeof()`
 - Commentaar:
 - Single line comments
 - Multi-line comments

De leerling kan een if-statement schrijven en kan deze uitbereiden met een else if en else.

De leerling weet wat functies en return values zijn, hij/zij kent de begrippen parameter en argument en weet hoe een functie met parameters geïmplementeerd wordt.

De leerling kent JavaScript build-in functions en kan hier mee werken.

- Behandelen huiswerk week 1
- If-statement:
 - If-statement
 - If-else-if statement
- Functions:
 - Function & functienamen en return values
 - Functions with parameters and arguments
- JavaScript build-in functions:
 - `concat()`
 - `prompt()`
 - `date()`

3	<p>De leerling kent het begrip variabele en de verschillende soorten variabelen.</p> <p>De leerling is bekend met het begrip scope en kan deze kennis toepassen bij het aanmaken van variabelen en het wijzigen ervan.</p> <p>De leerling kent de verschillende loops, is bekend met de termen break en continue en kan deze toepassen.</p> <p>De leerling kent het begrip switch en kan deze implementeren.</p> <p>De leerling is bekend met array functions, kent het verschil tussen JS functions en arrow functions. De leerling kan arrow functions met params implementeren.</p>	<ul style="list-style-type: none"> • Behandelen huiswerk week 2 • Variabelen & datatypen: <ul style="list-style-type: none"> ◦ (link) undefined ◦ var, const, let (default undefined) • Function: <ul style="list-style-type: none"> ◦ Scope ◦ Loops: <ul style="list-style-type: none"> ◦ for ◦ while ◦ do while ◦ break & continue ◦ Switch ◦ Arrow functions ◦ Arrow functions with params en arguments
4	<p>Eerste JavaScript II toets.</p> <p>De leerling is bekend met het begrip datastructuur en kan werken met arrays. Hiertoe behoren het toevoegen, opvragen van elementen.</p>	<ul style="list-style-type: none"> • Behandelen huiswerk week 3 • Eerste toets over stof van de voorgaande weken. • Toets bespreken • Datastructuren: <ul style="list-style-type: none"> ◦ Arrays: <ul style="list-style-type: none"> ◦ Element van array ◦ slice ◦ shift ◦ pop

5	De leerling is bekend met het begrip dictionaries en kan werken met key-value pairs.	<ul style="list-style-type: none"> • Bespreken huiswerk week 4 • Datatructuren: <ul style="list-style-type: none"> ◦ Dictionaries
6	De leerling is bekend met het begrip class, begrijpt waar een constructor voor wordt gebruikt. Hierbij komt het begrip scope terug.	<ul style="list-style-type: none"> • Bespreken huiswerk week 5 • Classes • Constructor • Scope (global variables)
7	<p>De voorgaande weken worden samengevat als voorbereiding op de toets.</p> <p>De leerling is bekend met de basis van JavaScript en kan zonder internet scripten.</p>	<ul style="list-style-type: none"> • Bepreken huiswerk week 6 • Samenvatten stof JavaScript II
8	Eindtoets period 3.	<ul style="list-style-type: none"> • Bespreken huiswerk week 7 • Eindtoets JavaScript II

Afhankelijk van het tempo wordt/worden (er) extra-curriculaire theorie/opdrachten toegevoegd aan de Python.

Begrippen

- Software developer = Softwareontwikkelaar = Een persoon die code schrijft met als doel software te ontwikkelt.

[B1] Integrated Development Environment (IDE) = Computerprogramma die de softwareontwikkelaar (software developer) ondersteunt bij het schrijven van code.

- Datastructuur = De manier waarom data word opgeslagen.

Bronnen

[1] Instaptoets: <https://b.socrative.com/login/student/>

Roomname: ROCVAAO

De instaptoets telt niet mee voor je eindcijfer!

[2] Les-opdrachten zijn anders dan het huiswerk en worden tijdens de les gemaakt. Van de leerling wordt verwacht dat hij/zij

vóór iedere les zowel de les-opdrachten als het huiswerk heeft gemaakt en ingeleverd bij de docent.

Les 1: Herhaling JavaScript I

Console

Tijdens de uitleg van JavaScript (JS) maakt de docent gebruik van de console van Google Chrome.

De console kun je openen door te klikken op de drie puntjes naast de adresbalk, deze opent een lijst van mogelijkheden. Click op "Meer hulpprogramma's" -> "Hulpprogramma's voor ontwikkelaars", merk op dat de console - afhankelijk van je instellingen - aan de rechterkant van je browser verschijnt.

De console kan ook geopend worden met de keyboard shortcut: `CTRL + SHIFT + I` of `fn + F12`

Variabelen

Een variabele verwijst naar een plekje in het computergeheugen. Met behulp van de variabele kun je van een bepaalde waarde uit het geheugen tonen/gebruiken en - als het nodig is - veranderen.

Om verwarring te voorkomen, is het belangrijk dat de variabele naam uniek is. Dit betekent dat de variabelenaam niet mag bestaan.

Waar moet een variabele naam aan voldoen?

1. Het belangrijkste is dat de naam van een variabele uniek moet zijn.
2. De naam van de variabele moet beginnen met een underscore(`_`), kleine- of hoofdletter.
3. Het eerstvolgende woord in de naam van de variabele mag beginnen met een letter, `_` of cijfer.

Variabele namen zijn case sensitive [**B1**], dit betekent dat er onderscheid wordt gemaakt tussen X en x.

Hoe kan ik een variabele aanmaken en gebruiken?

Het is mogelijk om een variabele eerst aan te maken en dan een waarde toe te kennen.
Bijvoorbeeld:

```
var nummer;  
nummer = 42;
```

Als je weet welke (begin)waarde een variabele heeft, kun je deze gelijk toekennen aan een variabele. Bijvoorbeeld:

```
var nummer = 42;
```

Code

JavaScript code kun je schrijven tussen een script tag, nog netter zou zijn om het in een external JavaScript file te schrijven.

Beide manieren worden hieronder gedemonstreerd:

<script>

De <script> tag wordt gebruikt om een client-side script (JavaScript) te definiëren en kan een script bevatten, of verwijzen naar een script.

Het volgende voorbeeld laat een HTML skeleton zien, met een script tag die JavaScript bevat:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8">  
    <title>Mijn prachtige titel</title>  
    <script type="text/javascript">  
      var stelling = "Minecraft is leuker dan Fortnite";  
      document.write(stelling);  
    </script>  
  </head>  
  <body>  
  
  </body>  
</html>
```

Het is ook mogelijk om JavaScript in een extern bestand te schrijven. Met behulp van de <script> tag kun je vervolgens verwijzen naar dit bestand. De bovenstaande code zou ook als volgt geschreven kunnen worden:

HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Mijn prachtige titel</title>
    <script src="gameStelling.js"></script>
  </head>
  <body>
    </body>
</html>
```

JavaScript (naam van het bestand is gameStelling.js):

```
var stelling = "Minecraft is leuker dan Fortnite";
document.write(stelling);
```

Je kunt JavaScript dus in je HTML script opnemen, of in een apart JavaScript bestand beschrijven. Afhankelijk van de manier zou je de `<script>`-tag moeten veranderen. Let hierbij op het volgende verschil:

```
<script type="text/javascript">Hier je script</script>
```

of

```
<script src="JavaScriptBestandNaam.js"></script>
```

Operators

Net als in de meeste programmeertalen, heeft ook JavaScript verschillende categorieën voor operators **[B2]**, namelijk:

- Vergelijkende operatoren
- Rekenkundige operatoren
- Logische operatoren

Vergelijkende operatoren

Vergelijkende operatoren worden gebruikt om waarden met elkaar te vergelijken en returnen een True/False. Deze waarden zijn boolean waarden. Het onderstaande tabel - Tabel 1 - beschrijft vergelijkende operatoren in JavaScript:

Operator	Beschrijving
==	Gelijk aan
!=	Niet gelijk aan
<	Kleiner dan
<=	Kleiner dan of gelijk aan
>	Groter dan
>=	Groter dan of gelijk aan

Tabel 1: Overzicht van vergelijkende operatoren

Rekenkundige operatoren

De rekenkundige operatoren kunnen gebruikt worden voor berekeningen. Voorbeelden van deze berekeningen zijn bijvoorbeeld optellen, aftrekken, vermenigvuldigen, delen, rest berekeningen en exponenten.

Tabel 1 toont verschillende rekenkundige operatoren die je kunt gebruiken wanneer je programmeert in JavaScript. Er zijn twee getallen als voorbeeld genomen om de syntax van een operator uit te leggen. Dit zijn de getallen 4 en 2.

Operator	Beschrijving	Syntax
+	Plus: telt twee waarden bij elkaar op.	4 + 2
-	Min: trekt twee waarden van elkaar af.	4 - 2
*	Vermenigvuldig: telt twee waarden herhaaldelijk bij elkaar op.	4 * 2
/	Delen: trekt twee waarden herhaaldelijk van elkaar af.	4 / 2
%	Modulo: berekend de restwaarde van een deelsom.	4 % 2

Tabel 2: Overzicht van rekenkundige operatoren

Lesopdracht 1: Hieronder staan enkele opgaven. Geef voor iedere regel aan wat het resultaat is en leg in je eigen woorden uit waarom dit zo is.

1) `5 < 3 != 4 > 8`

- 2) $18 \% 4$
- 3) $2 + 10 * 4$

Logische operatoren

Logische operatoren worden gebruikt met true/false-waarden en geven - afhankelijk van de gebruikte operator - een true of false waarde terug. Tabel 3 toont verschillende logische operatoren die je kunt gebruiken.

Operator	Beschrijving	Syntax
&&	Logische en	true && false
	Logische of	true false
!	Logische niet	!false

Tabel 3: Overzicht van logische operatoren

Lesopdracht 2: Hieronder staan enkele opgaven. Geef voor iedere regel aan wat het resultaat is (true/false) en leg in je eigen woorden uit waarom dit zo is.

- 4) true && true
- 5) true && false
- 6) true || false
- 7) !true

Datatypes

Datatypes **[B3]** worden ook wel gegevenstypes genoemd en zijn een belangrijk onderdeel van programmeren. Afhankelijk van een datatype kun je operators bijvoorbeeld wel/niet gebruiken.

Tabel 3 toont een overzicht met verschillende datatypes:

Datatype	Syntax	Variabelen van bepaalde datatypen
Strings	""	var str = "";
Numbers	getal (bijv. 1)	var nummer = 24;
Boolean	true/false	var booleanWaarde = true;

Lesopdracht 3: Stel dat we een variabele x hebben, die de som is van een string en nummer, bijvoorbeeld:

8) `var x = "Fifa " + 20`

Voer het bovenstaande stukje code uit in de console en gebruik de functie `typeof()` om de datatype van x te achterhalen. Leg uit welk datatype x heeft.

Comments

Comments kunnen gezien worden als aantekeningen en kunnen gebruikt worden om uit te leggen waarom code op een bepaalde manier wordt geschreven, of waarom een check belangrijk is. Een andere situatie waarin comments gebruikt kunnen worden is om collega's te waarschuwen om een bepaald bestand niet handmatig te veranderen.

JavaScript biedt de mogelijkheid om comments op een regel te schrijven (single line comments), of verdeeld over meerdere regels (multi-line comments). De volgende code snippet demonstreert beide opties:

```
// Dit is een single line comment
// Wanneer ik een dubbele forward-slash (//) gebruik, moet ik dit steeds opnieuw typen om een nieuwe regel als comment toe te voegen.

/*
Dit is een multi line comment. Ik kan hier zoveel text schrijven als ik wil, zonder dat ik extra tekens moet toevoegen.
*/
```

Lesopdrachten en huiswerk

De deadline voor de les - en huiswerk opdrachten kun je vinden in Microsoft Teams. De uitwerkingen kan je als Word - of text document inleveren in Microsoft Teams.

Het huiswerk voor deze week kun je vinden op Microsoft Teams; Team "Databases - `${classcode}`". Als je bijvoorbeeld in klas OITAOO9A zit, kun je je huiswerk vinden in Team "JavaScript- OITAOO9A".

Begrippen

[B1] Case sensitive = Hoofdletter gevoeling, met andere woorden, x en X zijn niet hetzelfde.

[B2] Operators = Een andere benaming voor be(werker). Een voorbeeld is bijvoorbeeld een rekenkundige operator +. Deze wordt gebruikt om op te tellen.

[B3] Datatype = Datatype, ook wel een gegevenstype genoemd. Voorbeelden zijn integers (getallen), string (text).

Les 2: For-lussen (For-loops)

Wanneer een bepaalde herhaling meerdere malen moet gebeuren, is het een goed idee om een lus **[B1]** te gebruiken. Zo een loop wordt uitgevoerd zodra er aan een bepaalde voorwaarde **[B2]** wordt voldaan. De for-lus is de meest compacte manier van het schrijven van een lus. De for-loop gaan we dan ook in deze les behandelen.

Vanaf nu gaan we te werk met de engelse termen. "Lus" noemen we voortaan een "loop". Dit betekend dat we de "for-lus" voortaan "for-loop" gaan noemen.

For-loop

De for-loop bestaat uit drie belangrijke onderdelen, namelijk:

- De initialization **[B3]**. Dit is de eerste stap wordt voor de iteratie **[B4]** uitgevoerd en geeft de beginwaarde aan.
- De voorwaarde. De voorwaarde bepaald hoe lang een for-loop moet lopen **[B5]**. De for-loop wordt uitgevoerd zolang de voorwaarde/conditie waar (true) is.
- De toenamefactor. Deze bepaald hoeveel er bij de ge-initialiseerde variabele opgeteld/afgetrokken moet worden.

Let op! De toenamefactor mag ook negatief zijn, dan is er namelijk sprake van een "afnamefactor".

De bovenstaande onderdelen worden in een for-loop op dezelfde regel geplaatst, waarbij ze gescheiden worden met een punt-komma.

Een put-komma wordt in het Engels een semicolon genoemd en bestaat uit een punt en een komma -> ;

Syntax

De syntax voor de for-loop is als volgt:

```
for(var variabelenaam = beginwaarde; variabele vergelijkende operator eindwaarde; variabele met toenamefactor){
```



```
// hier komt het script dat we willen herhalen
```

```
}
```

Laten we deze syntax uitwerken met een voorbeeld. Stel dat we achter elkaar tot en met 10 willen tellen, en dat we starten met het tellen bij 0. Dus: 0 1 2 3 4 5 6 7 8 9 10. Zoals je ziet, kunnen we dit achter elkaar schrijven. We kunnen dit echter ook laten doen door een for-loop.

Er zijn dus 3 stappen die we moeten uitwerken:

- We willen beginnen met tellen bij 0. Dus de beginwaarde (initialisatie) is 0: `var i = 0`.
- We willen tot en met 10 tellen, dus de 10 is inbegrepen en de loop moet dus stoppen voor we bij 11 komen: `i <= 10`.
- We beginnen met het tellen bij 0, dus na elke iteratie moeten we een optellen bij de variabele `i`: `i++`.

Let op! `i++` betekent dat we een variabele `i` hebben en dat we hier elke keer 1 bij optellen. We doen dus als het ware:

```
i = i + 1
```

Nu we de drie stappen hebben uitgewerkt, kunnen we verder met het schrijven van de for-loop:

```
for(var i = 0; i <= 10; i++){  
  document.write(i);  
}
```

Daar staat ie dan, onze eerste for-loop in JavaScript!

Lesopdrachten/huiswerk

Nu we weten hoe we een for-loop schrijven in JavaScript, kunnen we zelf aan het werk. Maak de volgende les-/huiswerk opdrachten en lever deze in via Microsoft Teams.

De volgende opdrachten kun je uitwerken in JSFiddle: <https://jsfiddle.net/>. Schrijf je script in de JavaScript-sectie en click linksboven op *Run* om je script te executen.

LET OP! JSFiddle hanteert een donker thema, waardoor de output in de *result*-sectie niet goed toont. Door deze regel onderaan je JS-editor toe te voegen, kun je de achtergrond kleur van je editor wit maken: `document.body.style.backgroundColor = "white";`

Les - / huiswerk opdracht 1: Schrijf een for-loop die tien keer loopt. Deze for-loop moet een halve kerstboom maken. De output moet er dus zo uit zien:

```
*  
**  
***  
****  
*****
```

Bij elke iteratie komt er dus 1 ster bij.

Tip: gebruik string concatenation.

Let op! Het gebruik van nested for-loops en repeat() is niet toegestaan.

Les - / huiswerk opdracht 2: Schrijf een for-loop die tien keer loopt. Deze loop moet aftellen van 10 tot en met 0.

Les - / huiswerk opdracht 3: Schrijf een for-loop die twintig keer loopt. Deze loop moet voor ieder getal aangeven of dit een even - of oneven getal is.

Voor even getallen print je *`${getal}` is een even getal*. Waarbij `${getal}` je variabele is.
Voor oneven getallen print je *`${getal}` is een oneven getal*.

`${getal}` is het getal van je variabele tijdens de iteratie.

Begrippen

[B1] Een *lus* wordt in het Engels een *loop* genoemd.

[B2] Een ander woord voor *voorwaarde* is een conditie. In het Engels wordt dit de *condition* genoemd.

[B3] *Initialization* is de Engelse term voor het zetten van een beginwaarde (initialiseren).

[B4] Iteratie is een ander woord voor *herhaling*. Een for-loop bestaan uit meerdere iteraties. De Engelse term hiervoor is *to iterate*.

[B5] In een for-loop wordt meerdere keren ge-looped. Het lopen wordt ook wel itereren genoemd. Zie ook begrip 4.

Bronnen

[1] JSFiddle: <https://jsfiddle.net/>

LEGENDA

Deze pagina dient als een legenda bij het begrijpen van de lessen. Hieronder zie je een overzicht van mogelijkheden:

Belangrijke informatie:

Belangrijke informatie kun je vinden in deze blauwe ballon.

Lesopdrachten:

Lesopdrachten moeten uitgewerkt en ingeleverd worden bij de docent en kun je herkennen aan deze oranje ballon.

Inline code:

Wanneer er in een zin een stukje code wordt getoond, kun je het herkennen aan de inline code ballon: `"Hello" + "world" + 1` .

Code snippet:

Wanneer een code snippet **[B1]** wordt gebruikt, kun je deze herkennen aan de grijze wol met een getal langs de kantlijn. Bijvoorbeeld:

```
print('Hallo!')
```

Begrippen:

Het kan zijn dat er woorden worden gebruikt die mogelijk extra uitleg nodig hebben. Deze worden aangeduid met een rechte haar, gevolgd door een B en dan een nummer. Het nummer bepaalt de volgorde van begrippen. Een voorbeeld hiervan is hierboven te zien, naast het woord "code snippet". Dit is het eerste begrip, daarom zie je een 1 naar de B.

De uitleg van "code snippet" kun je op dezelfde pagina onder titel "Begrippen" terug vinden:

Begrippen

[B1] Code snippet = een stukje code (van een groter geheel).

Bronnen:

Bronnen kun je herkennen aan getallen tussen rechte haken (**[1]**). Deze verwijzen naar bronnen in de bronnenlijst. De bronnenlijst is te vinden aan het einde van de pagina, onder het kopje 'Bronnen'.

Bronnen

[1] www.google.com (dit is een slecht voorbeeld!)

Week 1: Uitwerkingen les - en huiswerk opdrachten

Week 1

Lesopdracht 1: Hieronder staan enkele opgaven. Geef voor iedere regel aan wat het resultaat is en leg in je eigen woorden uit waarom dit zo is.

- 1) $5 < 3 \text{ != } 4 > 8$
- 2) $18 \% 4$
- 3) $2 + 10 * 4$

Uitwerkingen lesopdracht 1:

1. We "knippen" de vergelijking in een linker - en rechter helft. De linker helft heet de volgende vergelijking: $5 < 3$. Dit is false, want 5 is niet kleiner dan 3. De rechter helft is ook false want 4 is niet groter dan 8. Hier staat dus false != false . De uitkomst hiervan is false, want ze zijn wel gelijk aan elkaar.
2. De modulo (%) geeft de restdeling weer. $18 \% 4 = 2$ want 4 past maximaal 4 keer in 18: $4 * 4 = 16$ en $18 - 16 = 2$.
3. Computers rekenen volgens de wiskundige voorrangsregels. Dit betekent dat de vermenigvuldiging voor de optelling plaatsvindt: $10 * 4 = 40$, dus $2 + 40 = 42$.

Lesopdracht 2: Hieronder staan enkele opgaven. Geef voor iedere regel aan wat het resultaat is (true/false) en leg in je eigen woorden uit waarom dit zo is.

- 4) true \&\& true
- 5) true \&\& false
- 6) true || false
- 7) !true

Uitwerkingen lesopdracht 2:

- `true && true = true`
- `true && false = false`
- `true || false = true`
- `!true = false`

Lesopdracht 3: Stel dat we een variabele `x` hebben, die de som is van een string en nummer, bijvoorbeeld:

8) `var x = "Fifa " + 20`

Voer het bovenstaande stukje code uit in de console en gebruik de functie `typeof()` om de datatype van `x` te achterhalen. Leg uit welk datatype `x` heeft.

Uitwerking lesopdracht 3:

```
var x = "Fifa " + 20 // x = "Fifa 20"
typeof(x) // datatype van x is "string"
```

Huiswerk opdracht 1: Maak een variabele `y` aan en zorg dat deze, door middel van een vergelijking, een waarde van het type Boolean krijgt.

Uitwerking huiswerk opdracht 1: `var y = false` . Dit kan ook met vergelijksoperatoren: `var y = 5 < 6`

Huiswerk opdracht 2: Hieronder staan enkele opgaven. Werk deze opgaven uit in je console en schrijf het antwoord achter iedere opgave.

8. `7 > 8 != true`

9. `false == false`

Uitwerking huiswerk opdracht 2:

- `7 > 8 = false`, want 7 is kleiner dan 8. Hier staat dus `false != true`. En dit klopt, `false` en `true` zijn niet gelijk aan elkaar.
- `false == false` is `true`, want beide boolean waarden zijn gelijk aan elkaar.

Huiswerk opdracht 3: JavaScript heeft een `concat()` build-in functie. Leg uit wat deze functie doet en laat zien hoe je dit resultaat met het plus-teken kunt krijgen.

Uitwerking huiswerk opdracht 3: `concat()` is een ingebouwde JavaScript functie en wordt gebruikt voor concatenation van twee of meerdere strings/arrays. Voorbeelden:

```
// string concatenation kan met behulp van de +, bijvoorbeeld:
```

```
"Hello " + "world" = "Hello world"
```

```
// string concatenation met behulp van de concat() functie:
```

```
"Hello ".concat("world") = "Hello world"
```

Huiswerk opdracht 4: Leg uit wat je met vergelijkingsoperatoren kunt doen. Geef een voorbeeld.

Uitwerking huiswerk opdracht 4: Vergelijkingsoperatoren vergelijken twee waarden met elkaar en returnen - afhankelijk van de vergelijking - een waarde van het type Boolean.

Huiswerk opdracht 5: Leg uit wat een datatype en geef drie voorbeelden.

Uitwerking huiswerk opdracht 5: Een datatype is een gegevens type. Drie voorbeelden zijn:

- String
- Number
- Boolean

Huiswerk opdracht 6: JavaScript heeft een `typeof()` build-in functie. Leg uit wat deze functie doet en laat een voorbeeld zien met een datatype.

Uitwerking huiswerk opdracht 6: `typeof()` is een ingebouwde JavaScript functie. Deze functie kan gebruikt worden om de datatype van een gegeven/variabele te bepalen. Bijvoorbeeld: `typeof(2) = "number"`.

Week 2: Uitwerkingen les - en huiswerk opdrachten

Les - / huiswerk opdracht 1: Schrijf een for-loop die tien keer loopt. Deze for-loop moet een halve kerstboom maken. De output moet er dus zo uit zien:

```
*  
**  
***  
****  
*****
```

Bij elke iteratie komt er dus 1 ster bij.

Tip: gebruik string concatenation.

Let op! Het gebruik van nested for-loops en repeat() is niet toegestaan.

Uitwerking les - / huiswerk opdracht 1:

```
var ster = "" // variabele ster met een empty string  
for(var i = 0; i < 10; i++){  
  // overschrijf de waarde van variabele ster  
  ster = ster.concat('*');  
  
  // print de ster elke keer op een nieuwe regel:  
  document.write(ster);  
  document.write("<br />");  
  
}
```

Les - / huiswerk opdracht 2: Schrijf een for-loop die tien keer loopt. Deze loop moet aftellen van 10 tot en met 0.

Uitwerking les - / huiswerk opdracht 2:

```
for(var i = 10; i >=0; i--){  
  document.write(i);
```

```
document.write("<br />");  
}
```

Les - / huiswerk opdracht 3: Schrijf een for-loop die twintig keer loopt. Deze loop moet voor ieder getal aangeven of dit een even - of oneven getal is.

Voor even getallen print je *$\${getal}$ is een even getal*. Waarbij $\${getal}$ je variabele is.
Voor oneven getallen print je *$\${getal}$ is een oneven getal*.

$\${getal}$ is het getal van je variabele tijdens de iteratie.

Uitwerking les - / huiswerk opdracht 3: Deze opgave kan op twee manieren uitgewerkt worden, namelijk:

```
for(var i = 0; i < 20; i++){  
  if(i % 2 == 0){  
    document.write(i + ' is een even getal');  
  }else{  
    document.write(i + ' is een oneven getal');  
  }  
    
  // print elke keer op een nieuwe regel:  
  document.write("<br /> ");  
}
```

Een andere methode zou zijn om de if-statement te vervangen met `i % 2 == 1`, bijvoorbeeld:

```
for(var i = 0; i < 20; i++){  
  if(i % 2 == 1){  
    document.write(i + ' is een oneven getal');  
  }else{  
    document.write(i + ' is een even getal');  
  }  
    
  // print elke keer op een nieuwe regel:  
  document.write("<br /> ");  
}
```

Les 3: For-loop, Continue - en Break statement

Tijdens les drie gaan we aan de slag met for-loops en leren we de *continue* - en *break* statement gebruiken.

For-loop

Tijdens les 2 hebben we kennis gemaakt met de for-loop. Hierbij hebben we de syntax van de for-loop behandeld en hebben we gezien dat de for-loop uit drie onderdelen bestaat, namelijk:

- Een variabele met beginwaarde
- Een vergelijking, deze wordt ook wel eens een conditie genoemd
- Een toenamefactor.

Let op! De toenamefactor kan ook negatief zijn. Dan is er sprake van een afname!

Syntax van for-loop

De for-loop wordt aangeduid door een *for* gevolgd door ronde haakjes en een accolade **[B1]** die de for-loop opent. De syntax ziet er dan ook zo uit:

```
for(variabele met beginwaarde; conditie; toenamefactor){  
    // hier komt de code die de for-loop moet uitvoeren.  
}
```

Laten we de for-loop uitwerken aan de hand van een voorbeeld.

Stel dat we een for-loop hebben met een variabele *i*. Deze variabele heeft een beginwaarde van 0. De for-loop gaat 10 keer itereren **[B2]** en print bij elke iteratie de waarde van variabele *i*. Na elke iteratie neemt de variabele *i* met 1 toe:

```
for(var i = 0; i < 10; i++){  
    document.write("i is "+i);  
}
```

De bovenstaande for-loop eindigt wanneer variabele *i* gelijk is aan 9. Want met 0 erbij, heeft de for-loop dan in totaal 10 keer geloopt. **Let op!** De conditie is $i < 10$ en niet $i \leq 10$, dus *i* kan in de bovenstaande for-loop nooit 10 worden!

Continue statement

Soms komt het voor dat je binnen een for-loop een if-statement gebruikt. Deze kan, afhankelijk van de conditie, een bepaalde stap uitvoeren. Hier kan een continue statement wel handig zijn. Deze zorgt er namelijk voor dat de for-loop door blijft gaan tot de aangegeven conditie van de for-loop gelijk is aan *false*.

We leggen de continue statement uit aan de hand van het volgende voorbeeld:

Stel dat we een for-loop hebben met een variabele *x*. Deze variabele heeft een beginwaarde van 1 en blijft itereren tot en met 10. Bij elke iteratie neemt *x* met 1 toe. Deze for-loop bevat een if-statement. Deze checkt of *x* gelijk is aan de waarde van de gebruikers input. Indien deze gelijk zijn aan elkaar, wordt een *continue* statement gebruikt om de waarde over te slaan. In alle andere gevallen is de waarde van *x* ongelijk aan userInput en wordt deze waarde afgedrukt. Bijvoorbeeld:

```
/*
Vraag de gebruiker om een willekeurig getal op te noemen en vertaal deze van data type string naar number.
Sla dit getal op in een variabele genaamd userInput.
*/

var userInput = parseInt(prompt("Noem een willekeurig nummer tussen de 1 en de 10?"));

for(var x = 1; x <=10; x++){

    /* Vergelijk of x gelijk is aan het ingevoerde nummer van de gebruiker.
    □ De waarde van x moet "overgeslagen" worden als deze gelijk is aan de invoer van de gebruiker.
    In alle andere gevallen moet i geprint worden.
    */

    if(x == userInput){
        continue;
    } else {
        document.write("De waarde van x is "+x);
    }
}
```

Break statement

Soms wil je tijdens een for-loop uit de loop stappen door deze te stoppen. Dit kun je doen met behulp van de *break* statement.

Na de *break* statement houdt de loop dus op.

Stel dat we een for-loop hebben met een variabele *y*. Deze variabele heeft een beginwaarde van 0 en een eindwaarde van 10. Deze for-loop bevat een if-statement. Deze checkt of *y* gelijk is aan de waarde van de gebruikers input. Indien deze gelijk zijn aan elkaar, wordt een *break* statement gebruikt om de for-loop te stoppen. In alle andere gevallen print de for-loop de waarde van variabele *y*. Bijvoorbeeld:

```
var leeftijd = parseInt(prompt("Wat is je leeftijd"));

for(var y = 1; y <= 10; y++){
  // De for-loop stopt met lopen zodra y gelijk is aan de ingevoerde leeftijd. Deze "stop" wordt veroorzaakt door
  de break statement.
  if(y == leeftijd){
    break;
  }else{
    document.write("Je leeftijd is niet gelijk aan " + y);
  }
}
```

Samenvatting continue - en break statement

1. De continue statement zorgt ervoor dat een gegeven wordt overgeslagen tijdens het lopen.
2. De break statement zorgt ervoor dat een for-loop stopt met itereren.

Les - en huiswerk opdrachten

Les - en huiswerk opdracht 1: Schrijf een for-loop die bij 5 begint en tot en met 20 telt. Bij elke iteratie moet de for-loop het cijfer printen op een nieuwe regel.

Les - en huiswerk opdracht 2: Schrijf een for-loop die bij 10 begint en 20 keer loopd. Deze for-loop moet alleen even getallen printen. Gebruik continue om de oneven getallen over te slaan.

Les - en huiswerk opdracht 3: Als de onderstaande stappen uitgewerkt zijn, heb je een stukje code geschreven die om user-input vraagt en deze als nummer (int) opslaat in een variabele genaamd *leeftijd*. Deze variabele wordt gebruikt in een for-loop. De for-loop moet stoppen als het getal van de iteratie gelijk is aan de leeftijd die is ingevoerd door de gebruiker.

- Schrijf een script die vraagt om een user input (gebruik `prompt()`) en deze waarde opslaat in variabele `leeftijd`.

- Bepaal met behulp van de `typeof()` functie wat het datatype van variabele `leeftijd` is. Print de datatype in de console van je browser met behulp van `console.log()`.

- In de vorige stap heb je met behulp van de `typeof()` functie de datatype van variabele `leeftijd` achterhaald. Schrijf deze als een comment achter de `console.log()` functie die je hebt geschreven in stap 2.

- Zet `parseInt()` om de `prompt()` functie uit stap 1. Check met behulp van de `typeof()` functie wat de datatype van variabele `leeftijd` nu is. Comment deze datatype aan het einde van de regel waar ook je variabele `leeftijd` staat.

- Laat de code van de voorgaande stappen staan. Druk 2x op enter om een ruimte te creëren. We gaan nu een for-loop schrijven die bij 0 begint met tellen en tot 30 telt.

- Schrijf, binnen de for-loop uit stap 5, een if-statement die checkt of het getal van de huidige iteratie gelijk is aan de waarde die is opgeslagen in de `leeftijd` variabele. Als deze conditie `true` is, print je met behulp van `document.write("De ingevoerde leeftijd is gelijk aan" + i)` de loop ma deze print stoppen met tellen. Dit doe je met behulp van een `break`-statement.

Bronnen

[B1] Accolade is een gekrult haakje { en }. Onder accolade openen verstaan we het haakje { en andersom, met accolade sluiten bedoelen we }.

[B2] Itereren is een ander woord voor lopen.

Week 3: Uitwerkingen les - /huiswerk opdrachten

Les - / huiswerk opdracht 1: Schrijf een for-loop die bij 5 begint en tot en met 20 telt. Bij elke iteratie moet de for-loop het cijfer printen op een nieuwe regel.

Uitwerking les - / huiswerk opdracht 1:

```
for(var i=5; i<=20; i++){  
  document.write("i is gelijk aan " + i + "<br />")  
}
```

Les - / huiswerk opdracht 2: Schrijf een for-loop die bij 10 begint en 20 keer loopd. Deze for-loop moet alleen even getallen printen. Gebruik *continue* om de oneven getallen over te slaan.

Uitwerking les - / huiswerk opdracht 2:

```
for(var i=10; i<=30; i++){  
  if(i % 2 == 0){  
    document.write(i + " is een even getal <br/>")  
  }else{  
    continue  
  }  
}
```

Les - / huiswerk opdracht 3: Als de onderstaande stappen uitgewerkt zijn, heb je een stukje code geschreven die om user-input vraagt en deze als nummer (int) opslaat in een variabele genaamd *leeftijd*. Deze variabele wordt gebruikt in een for-loop. De for-loop moet stoppen als het getal van de iteratie gelijk is aan de leeftijd die is ingevoerd door de gebruiker.

1. Schrijf een script die vraagt om een user input (gebruik *prompt()*) en deze waarde opslaat in variabele *leeftijd*.

Uitwerking opdracht 3 stap 1: `var leeftijd = prompt("Wat is jouw leeftijd?")`

2. Bepaal met behulp van de `typeof()` functie wat het datatype van variabele *leeftijd* is. Print de datatype in de console van je browser met behulp van `console.log()`.

Uitwerking opdracht 3 stap 2: `console.log(typeof(leeftijd))`

3. In de vorige stap heb je met behulp van de `typeof()` functie de datatype van variabele *leeftijd* achterhaald. Schrijf deze als een comment achter de `console.log()` functie die je hebt geschreven in stap 2.

Uitwerking opdracht 3 stap 3: `console.log(typeof(leeftijd)) // string`

4. Zet `parseInt()` om de `prompt()` functie uit stap 1. Check met behulp van de `typeof()` functie wat de datatype van variabele *leeftijd* nu is. Comment deze datatype aan het einde van de regel waar ook je variabele *leeftijd*.

Uitwerking opdracht 3 stap 4: `var leeftijd = parseInt(prompt("Wat is jouw leeftijd?")) // number`

5. Laat de code van de voorgaande stappen staan. Druk 2x op enter om een ruimte te creëren. We gaan nu een for-loop schrijven die bij 0 begint met tellen en tot 30 telt.

Uitwerking opdracht 3 stap 5:

```
var leeftijd = parseInt(prompt("Wat is jouw leeftijd")) // number
console.log(typeof(leeftijd)) // string

for(var i=0; i<=30; i++){

}
```

6. Schrijf, binnen de for-loop uit stap 5, een if-statement die checkt of het getal van de huidige iteratie gelijk is aan de waarde die is opgeslagen in de *leeftijd*. Als deze conditie *true* is, print je met behulp van `document.write()` de volgende tekst:

De ingevoerde leeftijd is gelijk aan 17

Let op! De gebruiker heeft hier dus aangegeven dat hij/zij 17 jaar is.

Uitwerking opdracht 3 stap 6:


```
/*
```

Hieronder staat de volledige uitwerking van opdracht 3.

```
*/
```

```
var leeftijd = parseInt(prompt("Wat is jouw leeftijd")) // number
```

```
console.log(typeof(leeftijd)) // string
```

```
for(var i=0; i<=30; i++){
```

```
  if(i == leeftijd){
```

```
    document.write("De ingevoerde leeftijd is gelijk aan " + i)
```

```
    break
```

```
  }else{
```

```
    continue
```

```
  }
```

```
}
```

Week 5: JavaScript functions

huiswerk

1. Schrijf een functie die een parameter heeft. Deze functie moet checken of de parameter een (on)even nummer is.
Indien het getal een oneven getal is, moet deze geprint worden.
2. Schrijf een for-loop die van 0 tot en met 20 loopt. Deze for-loop moet de functie uit stap 1 aanroepen zodat deze wordt uitgevoerd. De variabele van de for-loop moet je als argument meegeven aan de functie uit stap 1.