

# Basis statische website

Realiseren

hoofdstuk

# 4

## Introductie Javascript





## Algemene informatie

Onderwerp	Introductie Javascript
Leerdoel(en)	<ol style="list-style-type: none"><li>1. De student kan in eigen woorden uitleggen wat het begrip scripting betekent.</li><li>2. De student kan benoemen wat het verschil is tussen ingesloten javascript en externe javascript.</li><li>3. De student kan in eigen woorden uitleggen wat een variabele is en kan eenvoudige variabele aanmaken met de hieronder genoemde datatypen.</li><li>4. De student kan bij het aanmaken van een variabele aangeven wat het datatype is van een waarde en kan dit testen door gebruik te maken van typeof.</li><li>5. De student kan een prompt en een alert scripten met en zonder gebruik van variabelen.</li><li>6. De student kan teksten wegschrijven naar de console en weet hoe deze teruggevonden kan worden binnen de Developer Tools.</li><li>7. De student kan teksten ophalen en wegschrijven naar een HTML-pagina door gebruik te maken van de standaardfuncties getElementById() en innerHtml().</li><li>8. De student kan een aantal eenvoudige standaardfuncties en eigenschappen voor strings toepassen zoals .length, .substring(), .toLowerCase() en .toUpperCase().</li><li>9. De student kan activiteitendiagrammen lezen en dit vertalen naar javascriptcode.</li><li>10. De student kan de vergelijkingsoperatoren ==, ===, !=, &gt;, &lt;, &gt;=, &lt;= toepassen.</li><li>11. De student kan een eenvoudige if/else-statement scripten.</li></ol>
Vereiste voorkennis	De student moet een basiskennis hebben over HTML waarbij de student in staat is om eenvoudige teksten te kunnen coderen in HTML. Deze kennis is opgedaan in hoofdstuk 2.
Kwalificatiedossier	<ul style="list-style-type: none"><li><input type="checkbox"/> B1-K1-W1: Plant werkzaamheden en bewaakt de voortgang</li><li><input type="checkbox"/> B1-K1-W2: Ontwerpt software</li><li><input checked="" type="checkbox"/> B1-K1-W3: Realiseert (onderdelen van) software</li><li><input checked="" type="checkbox"/> B1-K1-W4: Test software</li><li><input type="checkbox"/> B1-K1-W5: Doet verbetervoorstellen voor de software</li> <li><input type="checkbox"/> B1-K2-W1: Voert overleg</li><li><input type="checkbox"/> B1-K2-W2: Presenteert het opgeleverde werk</li><li><input type="checkbox"/> B1-K2-W3: Reflecteert op het werk</li></ul>



## Inhoudsopgave

Algemene informatie .....	2
Inhoudsopgave .....	3
Introductie .....	4
Inhoud .....	4
Wat is scripten? .....	4
Javascript koppelen .....	4
Ingesloten Javascript .....	4
Externe Javascript .....	5
Alert .....	5
Variabelen .....	6
Naamgeving variabelen .....	6
Datatypen .....	7
Datatypen bepalen .....	8
Prompt .....	9
Strings aan elkaar plakken (String concatenation) .....	11
Console .....	12
Foutmeldingen in de console .....	12
Schrijven naar de console .....	13
Testen in de console .....	13
Schrijven naar HTML .....	14
Standaard eigenschappen voor strings .....	17
.length .....	17
Standaard functies voor strings .....	17
.toUpperCase() .....	17
.toLowerCase() .....	18
.substring() .....	18
Parameters .....	19
Activiteitendiagram .....	19
Vergelijkingsoperatoren .....	21
If/else statement .....	21



## Introductie

De webpagina's die je in het vorige hoofdstuk gemaakt hebt zien er al een stuk beter uit omdat je ze hebt opgemaakt met CSS. In dit hoofdstuk ga je aan de slag met het toevoegen van interactie aan een HTML-pagina. Zoals je eerder al geleerd hebt gebruik je hiervoor Javascript. Dat is weer een andere techniek dan HTML of CSS.

## Inhoud

### Wat is scripten?

Een script is een verzameling van instructies die uitgevoerd kunnen worden om een bepaald doel te bereiken. Vergelijk het met een recept. Om een bepaald gerecht te kunnen maken dien je een aantal instructies te doorlopen. Als je al die instructies uitvoert heb je het doel bereikt, het gerecht is klaar! De hoeveelheid instructies die je nodig hebt om het doel te bereiken hangt af van de grootte of moeilijkheid van het doel.

Ingrediënten	Bereidingswijze
75 g boter	1. Verwarm de oven voor op 200 graden.
200 g pure chocolade	2. Snijd de boter en chocolade in blokkjes en stukjes.
150 ml kraanwater	3. Smelt de boter met kraanwater en zout in de pan.
1 mespunt zout	4. Voeg na het afkoelen 3 eieren toe aan het beslag.
100 g bloem	5. Met 2 natte lepels leg je bolletjes deeg op de bakplaat.
3 eieren	6. Bak 25 min in de oven en prik daarna gaatjes in de onderkant. Dan 5 min op de kop in de oven laten uitdrogen.
375 ml slagroom	7. Klop de slagroom met vanillesuiker en de versteviger stijf.
1 zakje vanillesuiker	8. Vul de spuitzak (lange mond) met slagroom en vul daarmee de soezen.
4 g slagroom-versteviger	9. Laat de slagroom en chocolade smelten in de pan.
	10. Bestrijk de soezen met het chocoladeglazuur.

Als je gaat scripten moet je altijd goed opletten dat je geen instructies vergeet. Sommige instructies zijn soms zo vanzelfsprekend dat je ze nogal snel vergeet.



Je kunt nu **Oefening 4.1** maken.

### Javascript koppelen

Net zoals bij CSS kun je Javascript ook op twee verschillende plaatsen schrijven. Het kan in de HTML-code zelf (ingesloten Javascript) of in een eigen Javascript bestand (externe Javascript).

#### Ingesloten Javascript

Bij ingesloten javascript plaats je de javascript code in de HTML-code door gebruik te maken van het HTML-element script.



```
<head>
  <link rel="stylesheet" href="stylesheet.css">
  <script>
    // Javascript code
    alert("Welkom");
  </script>
</head>
```

## Externe Javascript

Net zoals bij CSS is het echter overzichtelijker om Javascriptcode in een extern bestand te plaatsen. Dit doe je door het bestand op te slaan als een Javascript-bestand. Een javascript bestand herken je aan de extensie .js. Omdat het een nieuw bestand is dien je ook hier weer een module header bovenaan te plaatsen. Let op, een module header zet je altijd in commentaar. Als je meerdere regels commentaar wilt toevoegen in Javascript begin je met /\* en eindig je met \*/. Dat is hetzelfde als in CSS!

```
/*
Auteur:      <auteursnaam>
Aanmaakdatum: <dd-mm-jjjj>

<omschrijving inhoud>
*/

alert("Welkom");
```

Als je bovenstaande code in een extern bestand plaatst, moet je dit nog wel koppelen aan het HTML-bestand. Net zoals je dat ook gedaan hebt bij een CSS-bestand. Dit doe je in de HTML-code met het element script. Deze heb je ook al gebruikt bij de ingesloten Javascript. Nu plaats je er echter geen javascript code in, maar verwijst je naar het Javascript-bestand zelf.

```
<head>
  <title>
  </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="styles/stylesheet.css">
  <script src="scripts/script.js"></script>
</head>
```

## Alert

Hierboven heb je al een voorbeeld gezien van een alert. Een alert is voorbeeld van een instructie die je kunt geven in Javascript. We noemen dit in een script- of programmeertaal ook wel een statement. Een statement sluit je in Javascript af met een ;.

Statement

```
alert("Welkom");
```



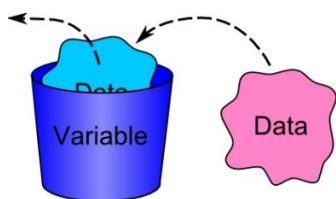
In de browser krijg je een pop-up venster met een opgegeven tekst. De opgegeven tekst moet je opgeven tussen de aanhalingstekens en ronde haken. Waarom dat precies is komt later nog aan bod.



Je kunt nu **Oefening 4.2** maken.

## Variabelen

In een variabele kun je waarden opslaan. Deze waarden kun je later weer gebruiken en/of aanpassen. Om de waarden makkelijk terug te vinden kun je een variabele een naam geven. Die naam is een verwijzing naar een locatie in het geheugen waar de waarde van de variabele wordt opgeslagen.



Elke programmeertaal maakt gebruik van variabelen. Variabelen kunnen waarden opslaan zoals naam, kleur, locatie. In Javascript maak je dat op de volgende manier aan:

```
var name = "Ralph";  
var color = "red";  
var location = "Den Bosch";
```

Variabelen maak je aan door ze te **declareren**.

```
var name = "Ralph";
```

Daarna geef je ze een waarde door ze te **initialiseren**.

```
var name = "Ralph";
```

Op internet zul je in plaats van var ook regelmatig **let** of **const** tegen komen. Wat deze precies doen gaan we later behandelen. Ben je nieuwsgierig kun je dit natuurlijk zelf al opzoeken!

## Naamgeving variabelen

Het allerbelangrijkste is dat je zinvolle namen bedenkt voor variabelen. Het moet duidelijk zijn wat je er in opslaat. Een variabele met de naam **variable1** zegt niet zoveel. Een variabele met de naam **age** wel. Daarnaast zijn er een aantal regels waar een naam van een variabele aan moet voldoen.

1. Een variabelenaam moet beginnen met een letter, een \$ of \_ (liggend streepje).



2. Het mag alleen bestaan uit letters, cijfers, \$ of een \_ (liggend streepje).
3. Een variabele naam is hoofdlettergevoelig. Dus **age** is niet hetzelfde als **Age**.
4. Je mag géén keywords gebruiken. Keywords zijn namen die in JavaScript al een andere betekenis hebben. Op de volgende bladzijde een overzicht van deze keywords.

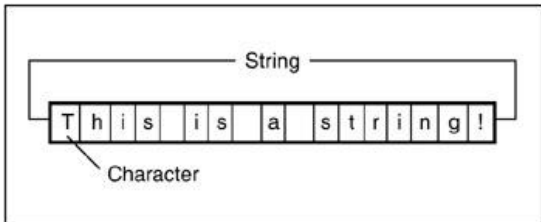
abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Words marked with\* are new in ECMAScript 5 and 6.

5. Volgens onze best practices gebruiken we Engelse namen voor de variabelen.

## Datatypen

Een datatype geeft aan wat voor soort data in de variabele wordt opgeslagen. De meest voorkomende datatypen kun je hieronder vinden.

Datatype	Omschrijving	Voorbeeld waarden
String	<ul style="list-style-type: none"><li>- Een reeks karakters.</li><li>- Een string zet je altijd tussen aanhalingstekens.</li></ul> 	"Dit is een string" "Koning Willem I College" "123" "true" "Voorbeeld 5"
Number	- Een geheel of decimaal getal.	1



	<ul style="list-style-type: none"><li>- Decimale getallen bevatten een punt in plaats van een komma.</li><li>- Een number zet je niet tussen aanhalingstekens. Als je dat doet wordt het een string. Met strings kun je niet gaan rekenen.</li></ul>	6523 123.50 0.78996 -1 -1.23 -9623.6543
Boolean	<ul style="list-style-type: none"><li>- Een variabele van het type boolean kent maar twee mogelijke waarden <b>true</b> of <b>false</b>.</li><li>- Een boolean zet je niet tussen aanhalingstekens. Als je dat doet wordt het een string. Je slaat dan de tekst "true" of "false" op.</li></ul>	true false

Javascript kent uit zichzelf een datatype toe op basis van de opgegeven waarde. Je hoeft dus van te voren niet op te geven welk datatype je wilt gebruiken. Bij veel andere talen moet dat echter wel!

## Datatypes bepalen

Je kunt in Javascript opvragen wat het datatype is van een variabele. Dat doe je door gebruik te maken van een **typeof**.

```
var name = "Ralph";  
alert(typeof name);
```

Als je dit uitvoert levert dat onderstaand resultaat op:



Nu kun je ook goed het verschil bepalen tussen een variabele met de waarde **123** en een variabele met **"123"** (met aanhalingstekens). Laten we eerst kijken naar wat er gebeurt als we variabelen met deze waarden in een alert plaatsen.

Javascript code	Resultaat
<pre>var exampleNumber = "123"; alert(exampleNumber);</pre>	
<pre>var exampleNumber2 = 123; alert(exampleNumber);</pre>	





Je ziet dat het resultaat precies hetzelfde is! Toch is er technisch gezien een verschil tussen deze twee. Het datatype is namelijk anders.

Javascript code	Resultaat
<pre>var exampleNumber = "123"; alert(typeof exampleNumber);</pre>	<div>127.0.0.1:55716 meldt het volgende</div> <div>string</div> <div>OK</div>
<pre>var exampleNumber2 = 123; alert(typeof exampleNumber);</pre>	<div>127.0.0.1:55716 meldt het volgende</div> <div>number</div> <div>OK</div>

Zoals bij de uitleg van datatype al is uitgelegd kun je met strings bijvoorbeeld niet rekenen, maar met numbers wel.

Javascript code	Resultaat
<pre>var exampleNumber = "123"; alert(exampleNumber + 1);</pre>	<div>127.0.0.1:55716 meldt het volgende</div> <div>1231</div> <div>OK</div>
<pre>var exampleNumber2 = 123; alert(exampleNumber2 + 1);</pre>	<div>127.0.0.1:55716 meldt het volgende</div> <div>124</div> <div>OK</div>

Je ziet nu dat als je gaat optellen bij een string het verkeerd gaat. Je ziet ook dat rekenen met een number wel gaat goed.



Je kunt nu **Oefening 4.3** maken.

## Prompt

Je hebt gezien dat je als programmeur een waarde van een variabele in de code kunt opgeven. Vaak wil je echter dat de gebruiker zelf een waarde kan opgeven en dat je die waarde later in je script nog een keer kan opvragen. Met een **prompt()** vraag je de gebruiker om een waarde in te voeren. De opgegeven waarde kun je vervolgens opslaan in een variabele.

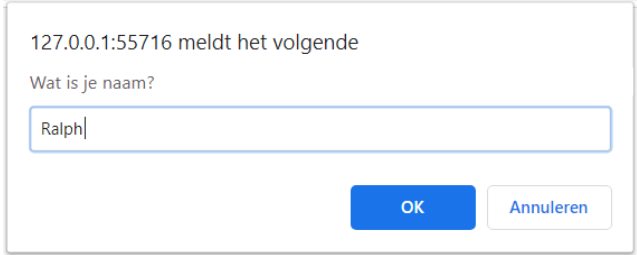
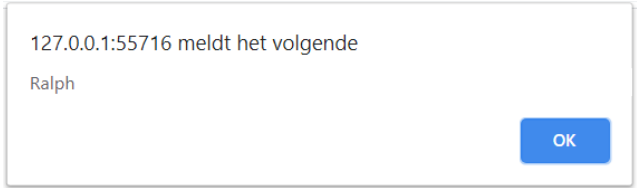
Javascript code	Resultaat
-----------------	-----------



<pre>var name = prompt("Wat is je naam?");</pre>	<div data-bbox="810 136 1449 392"><p>127.0.0.1:55716 meldt het volgende</p><p>Wat is je naam?</p><input data-bbox="836 241 1422 291" type="text"/><div data-bbox="1158 324 1422 369">OKAnnuleren</div></div>
--	--



Vervolgens kun je de waarde van die variabele bijvoorbeeld in een alert weergeven.

Javascript code	Resultaat
<pre>var name = prompt("Wat is je naam?"); alert(name);</pre>	<p>Na opstarten van de browser eerst de prompt:</p>  <p>Na invoeren van de naam en het klikken op OK komt de alert:</p> 

## Strings aan elkaar plakken (String concatenation)

In bovenstaand voorbeeld laat je in de alert alleen de naam zien. Meestal wil je een vaste zin laten zien met daarin de waarde van een variabele. Dat doe je door strings aan elkaar te plakken. Dat noemen we ook wel string concatenation. Dit doe je door gebruik te maken van het +-teken.

```
var name = prompt("Wat is je naam?");  
alert("Je naam is: " + name + ".");
```

Nu krijg je na het invoeren van de prompt onderstaande alert:



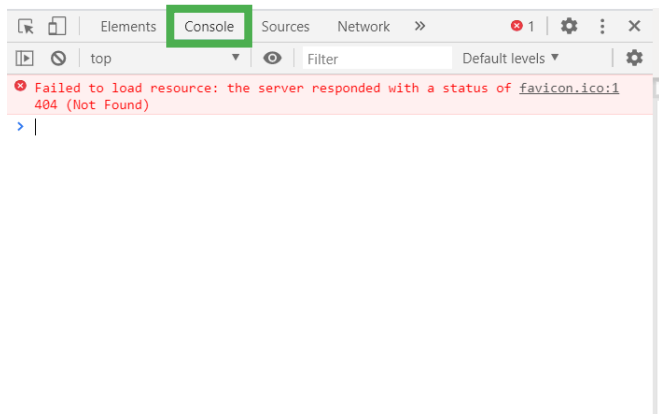
Het werkt in principe ook zonder gebruik van variabelen. In de praktijk kom je dat echter niet vaak tegen omdat je het dan net zo goed in één string kunt zetten.

```
alert("Dit is" + " een voorbeeld " + "van string concatenation.");
```



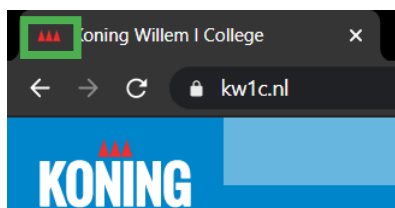
## Console

De console is een handig hulpmiddel in de browser om gemakkelijker fouten op te sporen, meldingen te schrijven die de gebruiker niet ziet maar de ontwikkelaar wel (logs), code te kunnen testen en nog veel meer. Je vindt de Console meestal door te drukken op F12. Hieronder een voorbeeld van de Console in Chrome.



## Foutmeldingen in de console

Je ziet dat er al een foutmelding in bovenstaand voorbeeld aanwezig is. Dit komt omdat de browser verwacht dat je een icon opgeeft voor je website. Een icon is een kleine afbeelding die in de titelbalk of het tabblad van de webbrowser wordt weergegeven.



Je mag deze foutmelding negeren, of je kunt het oplossen door een icon op te geven voor je website door onderstaande HTML-code toe te voegen aan je HTML-bestand.

```
<head>
  <title>
  </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="styles/stylesheet.css">
  <link rel="shortcut icon" href="images/favicon.ico">
  <script src="scripts/script.js"></script>
</head>
```

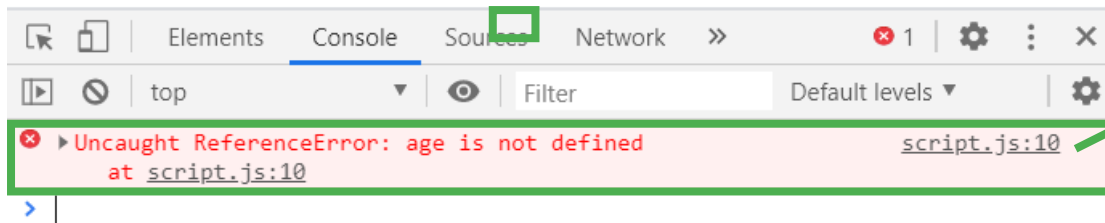
Hierbij is de waarde van het attribuut href uiteraard de locatie en bestandsnaam van de icon.

Onderstaande Javascriptcode levert ook een fout op. Er wordt namelijk een variabele in de alert gebruikt die niet gedeclareerd (aangemaakt) is.

```
var name = "Ralph";
alert(age);
```



In de console levert dit nu ook een fout op.



Hier zie je ook in welk bestand en welke regel (indicatief) de fout zit.

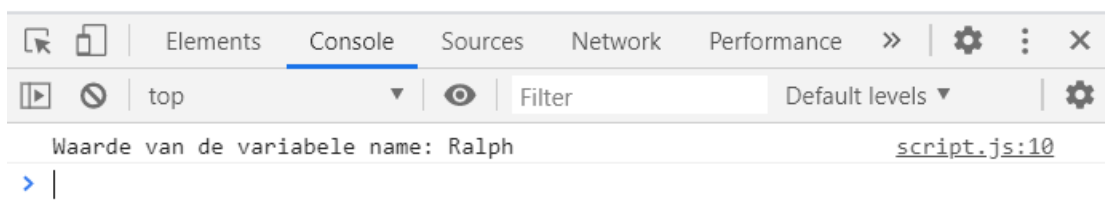
Als je code geschreven hebt die niet doet wat je verwacht, kijk dan altijd even of de console fouten aangeeft!

## Schrijven naar de console

De console kun je ook gebruiken om informatie weg te schrijven die je als programmeur wilt zien, maar die de gebruiker niet hoeft te zien. Bijvoorbeeld als je ergens in je code een bepaalde waarde van een variabele wilt weten. Je doet dit met een **console.log()**. In onderstaand voorbeeld kun je zien hoe je dat kunt doen.

```
var name = "Ralph";  
console.log("Waarde van de variabele name: " + name);
```

Resultaat in de console:



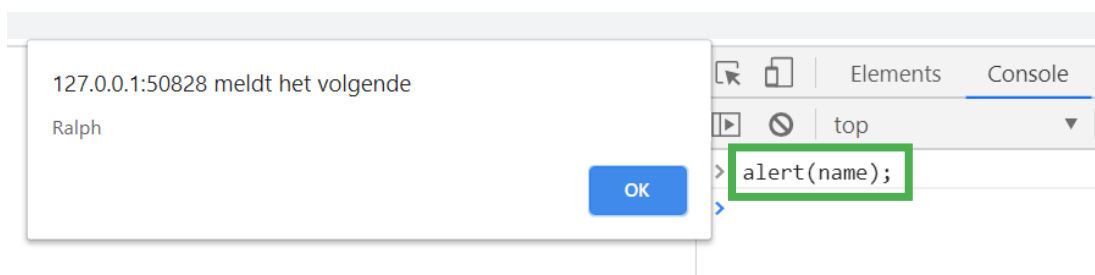
## Testen in de console

Naast fouten opsporen en informatie wegschrijven, kun je de console ook gebruiken om te testen. Je kunt in de console ook Javascript typen en laten uitvoeren.

Stel dat je in de Javascriptcode alleen een variabele name gedeclareerd en geïnitialiseerd hebt:

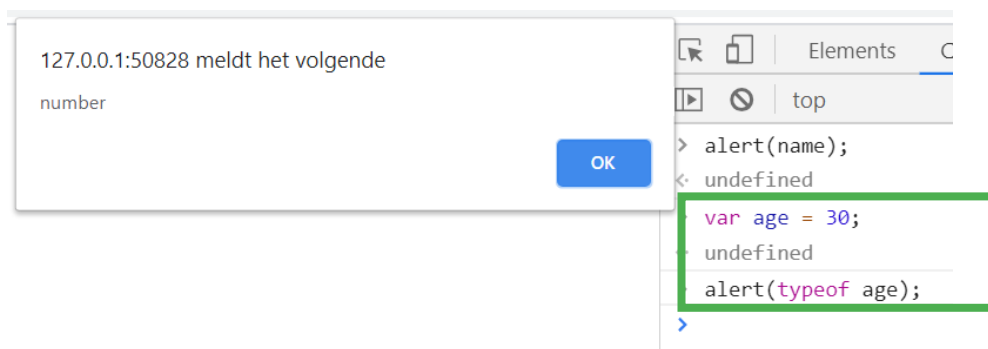
```
var name = "Ralph";
```

Je kunt nu via de console allerlei code uitvoeren om dingen uit te testen. Je doet dat door javascriptcode te typen achter de blauwe pijl die naar rechts wijst (als je Google Chrome gebruikt).





Je kunt er zelfs nieuwe variabele aanmaken om dingen te testen. Houd er wel rekening mee dat deze Javascriptcode niet wordt opgeslagen. Als je de browser refreshed is de code dus weer verdwenen. Het is alleen bedoeld om code te testen!



## Breakpoint

Een belangrijk onderdeel binnen de DevTools (F12) is het maken van een Breakpoint. Dit is een functionaliteit waarmee je je code vanaf een specifieke plek kunt pauzeren en stap voor stap kunt bekijken. Hiermee is het mogelijk om de waarden van variabelen uit te lezen op een specifiek moment tijdens het uitvoeren.

Om een breakpoint in een JavaScript-file te gebruiken doe je het volgende:

1. Open de webapplicatie.
2. Druk F12, tabblad Sources.
3. Links met de navigatie ga je naar het betreffende scriptfile: open deze. Je ziet de JavaScript in de Code Editor verschijnen.
4. Klik op of vóór het regelnummer waar je je code wil pauzeren. Er ontstaat een blauwe pijl op dat regelnummer. Dat is je breakpoint. Deze wordt dan ook rechts benoemd in het Debugger deel (incl. regelnummer: zie voorbeeld hierboven).
5. Voer je webapp uit (druk bijv. op F5).

Later tijdens de module Testen en Verbeteren gaan we dieper in op het gebruik van breakpoints.



Je kunt nu **Oefening 4.4** maken.

## Schrijven naar HTML

Je hebt nu geleerd hoe je teksten weg kunt schrijven naar een alert. Dat vinden de meeste bezoekers niet bepaald prettig. Daarnaast komt het niet echt professioneel over. Je weet nu ook hoe je teksten naar de console kunt schrijven. Echter is dit tekst die de bezoeker niet op de webpagina zelf ziet. Het zou veel netter zijn om een tekst weg te schrijven naar bijvoorbeeld een koptekst (h1-element in HTML). Dat is in Javascript mogelijk door gebruik te maken van **getElementById()** en **.innerHTML()**. Je ziet in getElementById() al een bekende HTML/CSS term, namelijk een id. Als je vanuit Javascript een tekst wilt wegschrijven door middel van de getElementById(), dien je het element een id te geven.

```
<h1 id="greet">  
</h1>
```

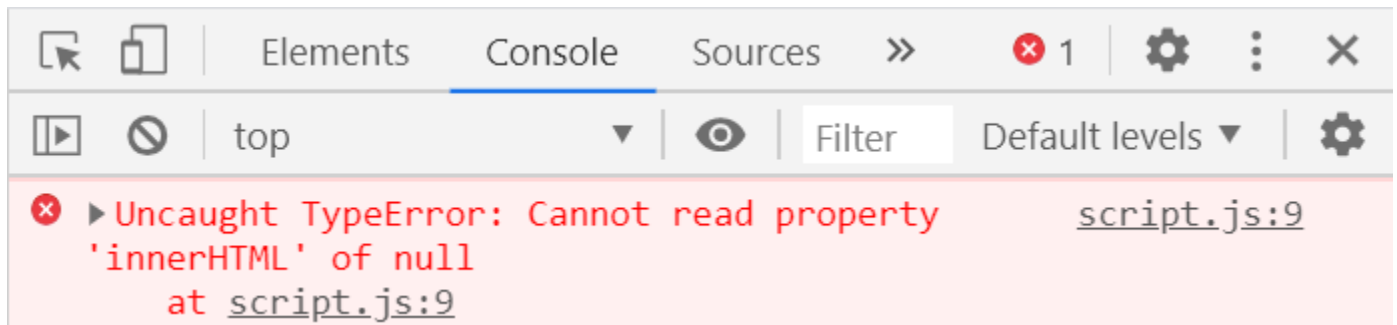


Vervolgens kun je in Javascript hier een tekst in plaatsen.

```
document.getElementById("greet").innerHTML = "Welkom";
```

Met de `getElementById` geef je aan welk element je wilt aanpassen. Met de `innerHTML` kun je vervolgens aangeven welke tekst je wilt laten zien in dat element.

Als je nu de webpagina gaat bekijken zie je dat er nog niets gebeurt. Dit komt doordat de console onderstaande foutmelding laat zien.





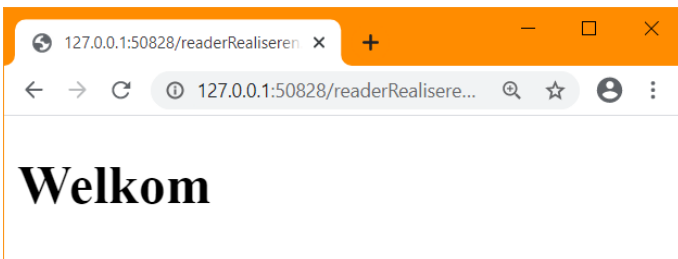
De console geeft aan dat hij de innerHTML niet kan lezen van iets dat niet bestaat (null). Dit komt omdat je Javascript koppelt in de head. De javascriptcode wordt dus al uitgevoerd voordat alle elementen op de webpagina worden geladen die in de body staan. De koptekst bestaat dus nog niet als de Javascript code wordt uitgevoerd.

Dit los je gemakkelijk op door het woord **defer** te gebruiken bij het koppelen van de javascriptcode. Hiermee zorg je ervoor dat eerst de gehele webpagina geladen moet worden, voordat de code uitgevoerd gaat worden.

```
<head>
  <title>
  </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="styles/stylesheet.css">
  <link rel="shortcut icon" href="images/favicon.ico">
  <script src="scripts/script.js" defer></script>
</head>
```

**Het is aan te raden om de defer standaard te gebruiken bij het koppelen van een Javascript-bestand.**

Als je de webpagina nu opnieuw gaat uitvoeren dan zal je zien dat de koptekst is gevuld met de tekst "Welkom".



Je kunt naast tekst ook HTML-code opgeven als string als waarde van een innerHTML.

```
document.getElementById("greet").innerHTML = "<em>Welkom</em> allemaal!";
```

Dit ziet er in de browser zo uit:



**Let op:** Zoals je geleerd hebt gebruik je een em-element om nadruk te geven aan een stuk tekst. Niet om deze dikgedrukt te maken. In het voorbeeld is gebruik gemaakt van Chrome. Chrome laat de tekst waar nadruk op ligt schuingedrukt zien. Het kan zomaar zijn dat een andere browser dat niet doet. Tekst schuingedrukt maken doe je in CSS met font-style: italic, en dus niet met een em-element!





## Standaard eigenschappen voor strings

Eigenschappen kun je zien als kenmerken van iets. Een stoel heeft bijvoorbeeld eigenschappen als hoogte, materiaalsoort en aantal poten.

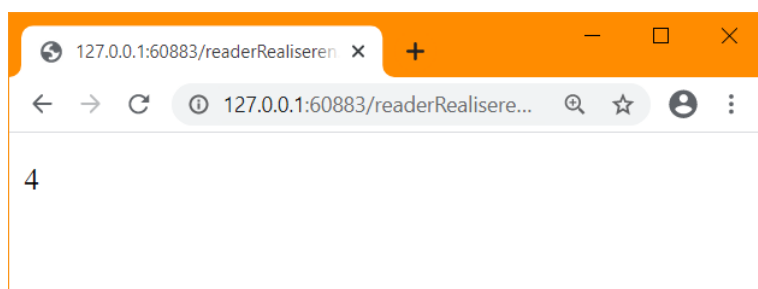
Een string heeft ook allerlei eigenschappen. Bijvoorbeeld het aantal letters van de string (de lengte).

### **.length**

Met de standaard eigenschap `.length` kun je het aantal karakters van een string ophalen.

```
var schoolnameLength = "KW1C".length;  
document.getElementById("result").innerHTML = schoolnameLength;
```

Dit ziet er in de browser zo uit:



## Standaard functies voor strings

Een functie is eigenlijk een stukje code die je een naam geeft, zodat je die code makkelijk later nog eens kan gebruiken zonder hem helemaal overnieuw te moeten schrijven. Nou kan je natuurlijk alles kopiëren en plakken, maar als je dan wat wil wijzigen moet dat op al die verschillende plekken en bovendien is het waarschijnlijk niet overzichtelijk. Omdat sommige stukken code zo vaak gebruikt worden heeft Javascript zelf ook al een aantal functies die je kunt gebruiken. Je herkent een functie in Javascript aan de `()`-tekens.

Je hebt dus al een aantal functies gebruikt, zonder dat je het misschien wist. Denk aan `alert()`, `prompt()`, `getElementById()` en `log()`. Er zijn ook functies die alleen bedoelt zijn om strings aan te passen. Denk daarbij aan omzetten naar hoofdletters of kleine letters, specifieke letters opzoeken of vervangen etc. Hieronder volgen een paar voorbeelden van functies waarmee je strings kunt aanpassen.

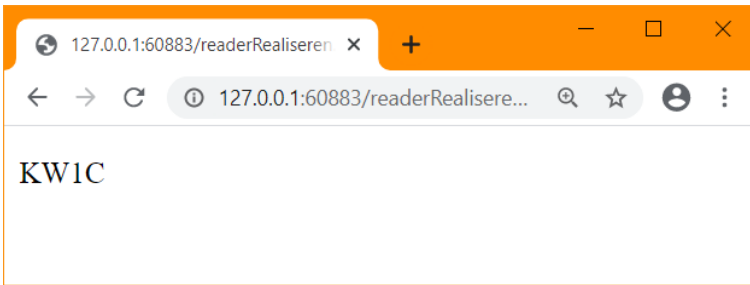
### **.toUpperCase()**

Met de standaard functie `.toUpperCase()` kun je een string omzetten naar hoofdletters.

```
var schoolname = "kw1c".toUpperCase();  
document.getElementById("result").innerHTML = schoolname;
```



Dit ziet er in de browser zo uit:

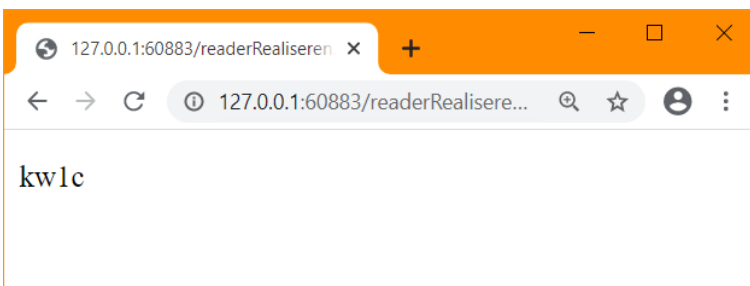


## **.toLowerCase()**

Met de standaard functie `.toLowerCase()` kun je een string omzetten naar kleine letters.

```
var schoolname = "KW1C".toLowerCase();  
document.getElementById("result").innerHTML = schoolname;
```

Dit ziet er in de browser zo uit:

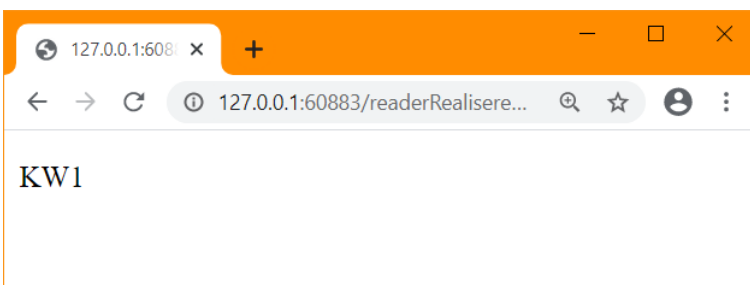


## **.substring()**

Met de standaard functie `.substring()` kun je een deel van een string ophalen.

```
var schoolname = "KW1C".substring(0,3);  
document.getElementById("result").innerHTML = schoolname;
```

Dit ziet er in de browser zo uit:



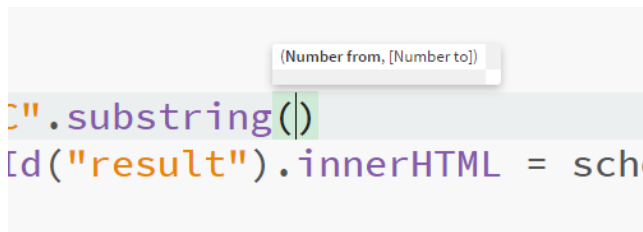
Je ziet dat er tussen de haakjes waarden zijn opgegeven. Dat noemen we parameters (zie uitleg in de volgende paragraaf). Je geeft met deze waarden aan bij welke letter je wilt starten en TOT welke letter je wilt doorgaan. De computer telt hierbij vanaf 0.

0	1	2	3
K	W	1	C

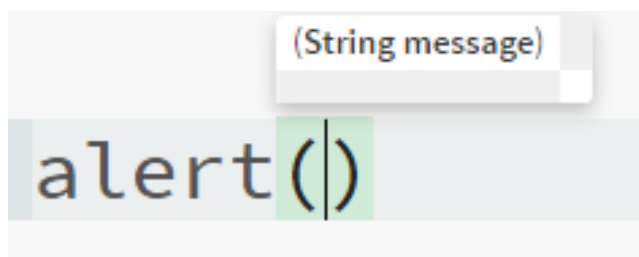


## Parameters

Je ziet dat er tussen de ()-tekens van de functie `substring` waarden zijn opgegeven. Dit noemen we **parameters**. Bij sommige functies kun je extra informatie meegeven die nodig is om de functie te kunnen uitvoeren. Als je een deel van een string wilt ophalen moet je bijvoorbeeld wel aangeven op welke positie je wilt beginnen en welke positie je wilt eindigen. Daar staan dan ook de cijfer 0 en 3 voor binnen de haakjes. Als je code in de code editor typt wordt je geholpen bij het invoeren van eventuele parameters. Zodra je het ()-teken plaats achter de naam van de functie zie je in een pop-up de mogelijke parameters staan. Zo weet je precies welke informatie de functie van jou wilt hebben.



Parameters heb je ook al vaker gebruikt bij onder andere de `alert()`, `prompt()`, en `log()`.



Als je een functie wilt gebruiken die geen parameters verwacht, zie je in die pop-up staan dat er geen parameters verwacht worden.



Je kunt nu **Oefening 4.5** maken.

## Activiteitendiagram

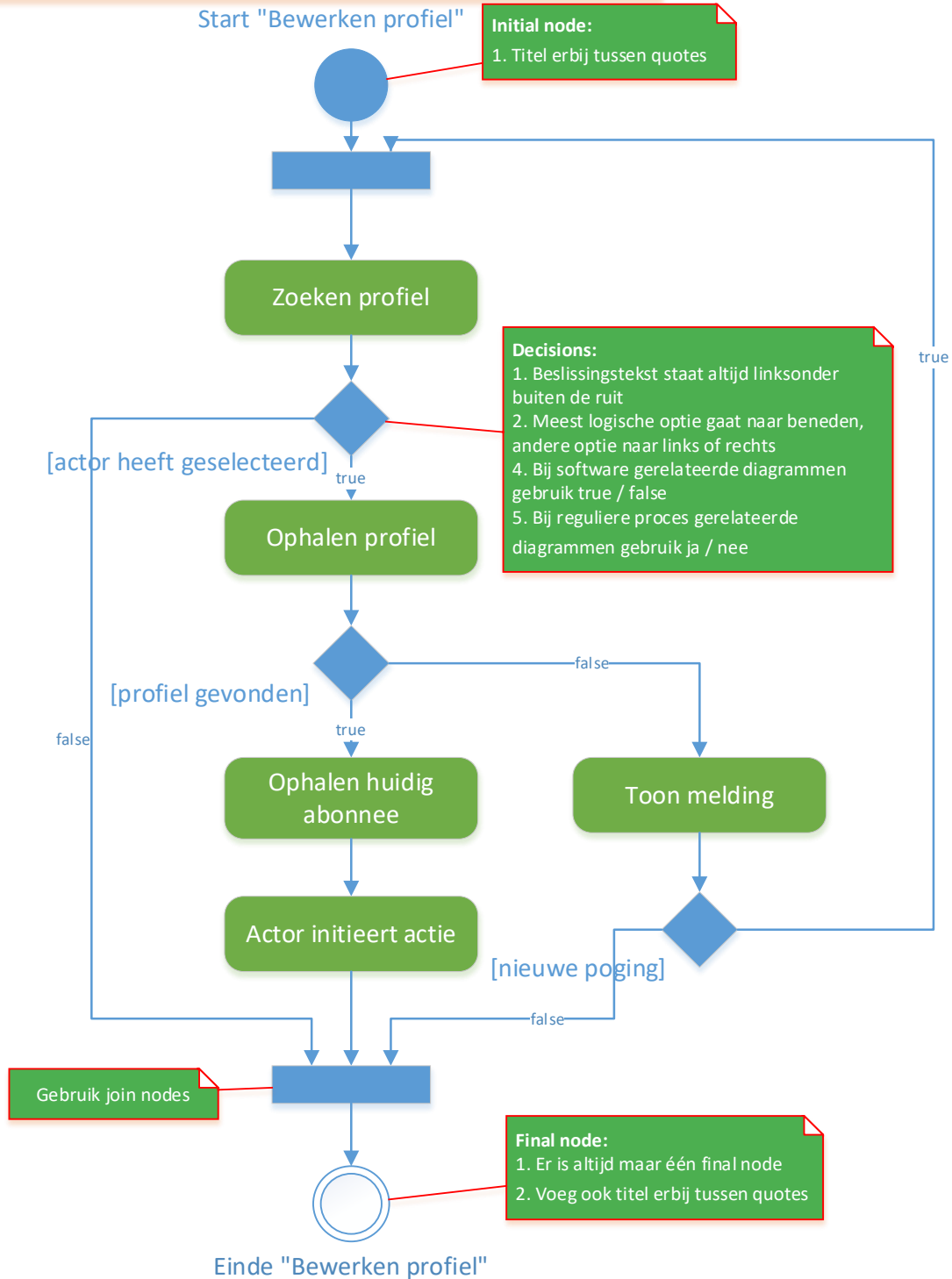
Bij het ontwerpen van een applicatie is het handig om activiteitendiagrammen op te stellen. Met een activiteitendiagram breng je in beeld wat het gedrag is van een applicatie. Welke instructies moeten er worden uitgevoerd op welk moment?

Tot dusver heb je alleen code geschreven waarbij alle regels code worden uitgevoerd. In de praktijk komt het echter vaak voor dat er bepaalde voorwaarden worden gesteld. Denk bijvoorbeeld aan een leeftijdschecker. Als de bezoeker jonger is dan een gestelde leeftijd moet er wat anders gebeuren dan dat de bezoeker wel een minimum leeftijd heeft. Deze keuzes en bijbehorende acties kun je goed weergeven in een activiteitendiagram.



### Algemene afspraken:

- Gebruik Visio Shape genaamd "UML Activity" (Engels) of "uml-activiteitendiagram" (Nederlands)
- Werk altijd van boven naar beneden
- Gebruik de standaard Visio layout
- Maak document nooit groter dan één A4 formaat





## Vergelijkingsoperatoren

Vergelijkingsoperatoren kunnen gebruikt worden om waarden met elkaar te vergelijken. Bijvoorbeeld in het verhaal van de leeftijdschecker. Je wilt dan controleren of de ingevoerde leeftijd gelijk is aan / hoger is dan de minimum leeftijd.

Dat zou je op onderstaande manier kunnen controleren.

```
var age = prompt("Hoe oud ben je?");  
var ageCheck = age >= 18;  
document.getElementById("result").innerHTML = ageCheck;
```

De tweede regel levert een waarde op van het type boolean (true/false). Als je nu 15 invoert, wordt in de paragraaf met id "result" false weergegeven. Als er 18 of hoger ingevoerd wordt er true weergegeven. Hieronder zie je de verschillende soorten vergelijkingsoperatoren en hun betekenis.

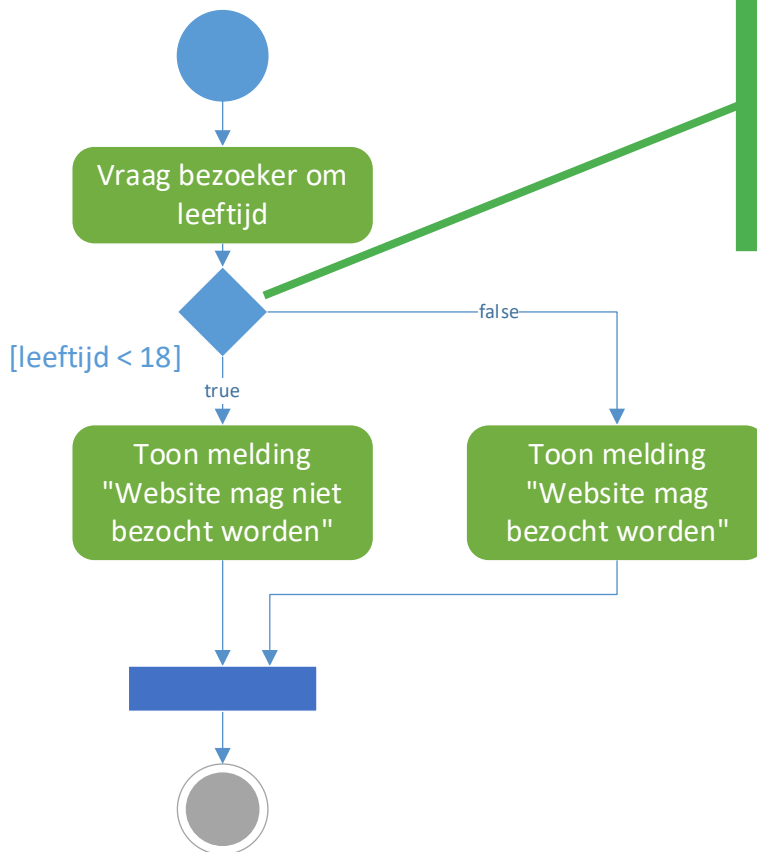
Vergelijkingsoperator	Betekenis	Voorbeeld	Resultaat
==	Gelijk aan	1 == 1 "1" == 1	True True
!=	Niet gelijk aan	1 != 2 1 != 1	True False
===	Identiek aan (het datatype moet hetzelfde zijn)	"1" === "1" "1" === 1	True False
!==	Niet identiek aan (het datatype moet anders zijn)	1 !== "1" 1 !== 1	True False
>	Groter dan	5 > 1	True
<	Kleiner dan	1 < 3	True
>=	Gelijk aan, of groter dan	5 >= 1 5 >= 5	True True
<=	Gelijk aan, of kleiner dan	1 <= 3 3 <= 3	True True

## If/else statement

In de vorige paragraaf heb je kunnen zien hoe je twee waarden met elkaar kunt vergelijken. De uitkomst daarvan is altijd true of false. In werkelijkheid wil je dat er bijvoorbeeld een andere melding wordt weergegeven. Dat kun je doen door een keuze in je programma in te bouwen. Een keuze kun je scripten door middel van een if/else statement. Op de volgende bladzijde zie je een eenvoudig activiteitendiagram met een keuzemoment.



Start "Controleren leeftijd"



Einde "Controleren leeftijd"

Bovenstaande activiteitendiagram kan op de volgende manier worden omgezet naar Javascript code.

```
// Vraag bezoeker om leeftijd
var age = prompt("Hoe oud ben je?");
// Als age onder de 18 is
if(age < 18)
{
    // Dan wordt deze code uitgevoerd
    document.getElementById("result").innerHTML = "Website mag
niet bezocht worden";
}
else
{
    // Anders wordt deze code uitgevoerd
    document.getElementById("result").innerHTML = "Website mag
bezocht worden";
}
```



```
if(conditie)
{
}
else
{
}
}
```

In de conditie kun je waarden met elkaar vergelijken waarbij je gebruik maakt van de vergelijingsoperatoren.

Bijvoorbeeld:

<code>name == "Ralph"</code>	Lees: ALS name gelijk is aan "Ralph"
<code>age &lt;= 17</code>	Lees: ALS age kleiner of gelijk is aan 17
<code>name.length &gt; 10</code>	Lees: ALS het aantal letters van name groter is dan 10

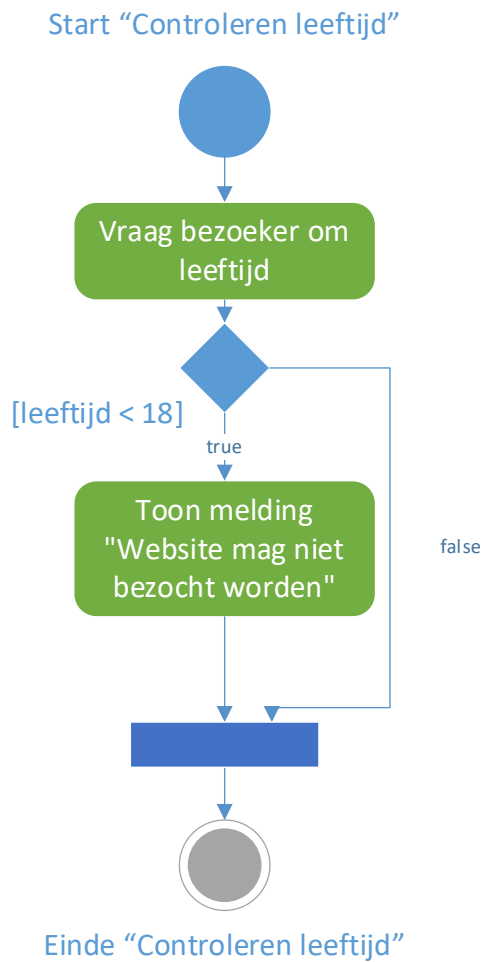
Code die wordt uitgevoerd als de conditie true oplevert.

Code die wordt uitgevoerd als de conditie false oplevert.

Als je de woorden vertaalt helpt dat misschien met het beter begrijpen van de code.

```
// Vraag bezoeker om leeftijd
var age = prompt("Hoe oud ben je?");
// Als age onder de 18 is
if(age < 18) ALS age <= 18
{
    DAN laat melding zien dat website niet bezocht mag worden
    document.getElementById("result").innerHTML = "Website mag
    niet bezocht worden";
}
else
{
    ANDERS laat melding zien dat website bezocht mag worden
    document.getElementById("result").innerHTML = "Website mag
    bezocht worden";
}
```

Je kunt er ook voor kiezen om geen else te gebruiken. In dat geval laat je alleen een specifieke code uitvoeren als er aan een bepaalde voorwaarde wordt voldaan, maar niet als daar niet aan wordt voldaan.



Je kunt nu **Oefening 4.6** maken.