1 Statements

Een computerprogramma bestaat uit een lijst van instructies die moeten worden uitgevoerd door een computer. Deze instructies noemen we 'statements'.

Javascript statements worden opgebouwd uit:

- Waardes
- Operatoren
- Expressies
- Sleutelwoorden
- Commentaren

Een statement mag over meerdere coderegels worden verdeeld. Hierbij worden regelovergangen en inspringen genegeerd. Om een overzicht te bewaren, moet elk statement worden afgesloten met een puntkomma.

JS-code

```
document.getElementById("inhoud").innerHTML = "Hallo";
```

Javascript is ***hoofdlettergevoelig*** (case sensitive). Het foutief gebruiken van hoofdletters en kleine letters in gereserveerde woorden of eigen functies en variabelen, kan leiden tot problemen bij het uitvoeren van de code in de browser.

JS-code

```
FOUT: document.getelementbyid("inhoud").InnerHTML = "Hallo";
FOUT: document.getelementbyid("inhoud").innerhtml = "Hallo";
FOUT: document.GetElementByID("Inhoud").innerhtml = "Hallo";

JUIST: document.getElementById("inhoud").innerHTML = "Hallo";
```

2 Variabelen en constanten

2.1 Variabelen

Met variabelen kunnen we bepaalde gegevens onthouden om ze nadien opnieuw te gebruiken. Variabelen hebben altijd een naam, ook wel identifier genoemd. Nadat we een variabele gedeclareerd hebben, refereren we naar de waarde van die variabele door de naam te gebruiken.

JS-code

```
// Declaratie en initialisering in één regel schrijven
let aantal = 1;

// Declaratie en initialisering verspreiden
let aantal;
aantal = 1;
```

Bovenstaande twee mogelijkheden doen allebei hetzelfde; ze maken eerst een variabele aan met de naam 'aantal' en geven deze variabele daarna de waarde '1'.

De waarde van een variabele kan doorheen de code gewijzigd worden, maar de naam van de variabele blijft altijd hetzelfde. ***Bij het kiezen van een naam moet je volgende regels in acht nemen:***

- Het eerste karakter moet een letter of underscore (_) zijn.
- De naam mag zowel hoofdletters als kleine letters bevatten.
- De naam mag letters, getallen, underscore en dollartekens bevatten.
- De naam mag geen spaties of andere lees- of speciale tekens bevatten.
- De naam mag geen gereserveerd woord (gekend woord in de webtaal) zijn.

- *JS-code*

```
// Juist
let getal;
let i;
let _totaal;
let aantalKoekjes;
let aantal_snoepjes;

// Fout
let 1cadeau;
let aantal-repen;
let break;
let snoepjes&koekjes;
```

2.2 Constanten

Een constante is een variabele waarbij de waarde niet aangepast kan worden na toekenning.

JS-code

```
// Declaratie en initialisering in één regel schrijven
const aantal = 1;

// Meerdere declaraties en initialisaties op één regel
const een = 1, twee = 2, drie = 3;
```

2.3 Gereserveerde woorden

Volgende woorden mogen we niet gebruiken als naam voor onze variabelen en constanten.

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	synchornized	switch	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

3 Gegevenstypes

Javascript kan heel wat verschillende gegevenstypes verwerken. We kunnen de verschillende gegevenstypes in twee groepen verdelen.

1. ***Primitieve gegevenstypes (primitive datatypes)***
    - Getallen (numbers)
    - Tekenreeksen (strings)
    - Logische waarden (booleans)
    - Null
    - NaN (not a number)
    - Undefined

2. ***Complexe- of objecttypen (reference datatypes en object datatypes)***
    - Array
    - Function
    - Object

Javascript is een 'loosely of dynamically typed language'. In Javascript moeten we uitsluitende de naam van de variabele opgeven. De conversie tussen de gegevenstypes verloopt automatisch.

3.1.1 Getallen

In Javascript kunnen we zowel gehele als gebroken getallen gebruiken.

JS-code

```
let number = 13;
let decimal = 3.14;
let avagadro = 6.0221e23;
```

Een browser geeft een ingetypte waarde (bvb. uit een formulier) altijd als een string terug. Alvorens we het getal kunnen gebruiken in onze javascript-code, moeten we het omzetten naar een getal.

JS-code

```
// parseInt() zet een string om naar een geheel getal.
// parseFloat() zet een string om naar een gebroken getal.
let getal1 = parseInt(prompt('Vul een geheel getal in', 'getal 1'));
let getal2 = parseFloat(prompt('Vul een gebroken getal in', 'getal 2'));


let totaal = getal1 + getal2
let tekst = getal1 + ' plus ' + getal2 + ' = ' + totaal;
alert(tekst);
```

3.1.2 Tekenreeksen

Strings zijn reeksen van tekens (ASCII-karakters). In Javascript worden strings tussen enkele of dubbele aanhalingstekens geplaatst.

JS-code

```
let zin = "Deze zin is een string";
```

Naast numerieke of alfanumerieke tekens kunnen strings ook speciale tekens bevatten, waaronder regelteruglopen, tabs en andere niet-afdrukbare tekens. Deze speciale tekens moeten we vooraf laten gaan door een backslash (\).

JS-code

```
alert("Dit is de eerste regel\nDit is de tweede regel");
alert("Dit stukje staat voor een tabstop\tDit stukje na een tabstop);
alert("Mijn favoriete quote is: \"Hakuna Matata\".");
alert("Installeer de juiste driver in C:\\Windows\\System.");
```

Via het gebruik van stringfuncties kunnen we informatie verkrijgen over een stringwaarde. Volgende functies zijn slechts een greep uit het volledige aanbod van stringfuncties.

JS-code

```
let welkom = "hallo iedereen";
let uitkomst;

// Hoeveel karakters bevat de string? 14
uitkomst = welkom.length;
alert(uitkomst);

// Welk karakter staat op plaats 0 (telling begint vanaf 0!)? h
uitkomst = welkom.charAt(0);
alert(uitkomst);

// Wat zijn de eerste vijf karakters? hallo
uitkomst = welkom.substring(0,5);
alert(uitkomst);

// Op welke plaats staat karakter 'i'? 6
uitkomst = welkom.indexOf("i");
alert(uitkomst);

// Op welke plaats staat karakter 'y'? -1 (komt niet voor in de string)
uitkomst = welkom.indexOf("y");
alert(uitkomst);

// Op welke plaats komt karakter 'e'het laatste voor? 12
uitkomst = welkom.lastIndexOf("e");
alert(uitkomst);

// Vervang karakter 'h' met karakter 'H. Hallo iedereen
uitkomst = welkom.replace("h", "H");
alert(uitkomst);

// Zet alle karakters in hoofdletters om. HALLO IEDEREEN
uitkomst = welkom.toUpperCase();
alert(uitkomst);

// Zet alle karakters in kleine letters om. Hallo iedereen
uitkomst = welkom.toLowerCase();
alert(uitkomst);
```

3.1.3 Logische waarden

Vergelijkingen, functies of lussen kunnen in Javascript een booleaanse waarde teruggeven. Een booleaanse waarde is waar (true) of onwaar (false).

JS-code

```
let getal = 30;

// Variabele 'getal' is gelijk aan 30. True
getal == 30;

// Variabele 'getal' is kleiner dan 31. True
getal < 31;

// Variabele 'getal' is groter dan 31. False
getal > 31;

// Variabele 'getal' is gelijk aan 31. False
getal == 31;
```

3.1.4 Null

De waarde 'Null' staat letterlijk voor 'niets'. Het is de waarde van een niet-gedeclareerde variabele.

3.1.5 NaN

De waarde NaN staat voor 'not a number'. Deze waarde krijgen we terug indien het programma een nummer had verwacht, maar deze niet heeft gekregen.

3.1.6 Undefined

De waarde 'undefined' geeft aan dat een variabele nog niet geïnitialiseerd is of een functie niets retourneert.

JS-code

```
let getal;
alert(getal);
```

4 Operatoren

Operatoren hebben we in het vorige hoofdstuk al gebruikt. Met operatoren kunnen we de variabelen in een script optellen, aftrekken vergelijken of overige handelingen mee uitvoeren.

We kunnen operatoren in verschillende categorieën indelen:

- Toewijzingsoperatoren (assignment operators)
- Wiskundige operatoren (arithmic operators)
- Stringoperatoren (string operators)
- Logische operatoren (logical operators)
- *Bitoperatoren (bitwise operators)*
- Vergelijkingsoperatoren (comparison operators)
- speciale (overige) operatoren

4.1 Toewijzingsoperatoren

Het isgelijkteken (=) is een toewijzingsoperatoren en gebruiken we voor:

- het toewijzen van waarden aan variabelen;
- het wijzigen van waarden van variabelen.

JS-code

```
// Kent de waarde 1 toe aan variabele getal1
getal1 = 1;

// Telt 1 op bij de waarde van getal1
getal1 = getal1 + 1;

// Telt 1 op bij de waarde van getal2
getal1 = getal2 + 1;

// stuurt de waarde van getal1 en getal2 door naar de functie optellen()
en slaat de geretourneerde waarde op in uitkomst
uitkomst = optellen(getal1, getal2)
```

4.2 Wiskundige operatoren

Met de wiskundige operatoren kunnen we wiskundige bewerkingen uitvoeren op numerieke waarden.

Operator	Uitleg
+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
%	Restwaarde berekenen
++	Getal plus 1
--	Getal min 1

JS-code

```
// Uitkomst is 15.
let uitkomst = 10 + 5;

// Uitkomst is 5.
let uitkomst = 10 – 5;

// Uitkomst is 50.
let uitkomst = 10 * 5;

// Uitkomst is 2.
let uitkomst = 10 / 5;

// Uitkomst is 1.
let uitkomst = 11 % 5;

// Uitkomst is 2.
let uitkomst = 1;
uitkomst++;

// Uitkomst is 5.
let uitkomst = 6;
uitkomst--;
```

4.3 String operatoren

Het plusteken (+) is een concatenatieoperator. Met deze operator kunnen we twee of meer strings samenvoegen tot één string.

JS-code

```
let zin = "Dit is  " + "een zin.";

let deel1 = "Dit is";
let deel2 = "een zin";
let uitkomst = deel1 + " " + deel2;
```

4.4 Logische operatoren

Met logische operatoren kunnen we nagaan of een expressie, bestaande uit logische waarden, waar of onwaar is.

Operator	Uitleg
&&	*Logische 'en'* Deze operator geeft 'true' terug indien alle expressies 'true' zijn. Deze operator geeft 'false' terug indien één of alle expressies 'false' zijn.
||	*Logische 'of'* Deze operator geeft 'true' terug indien één of alle expressies 'true' zijn. Deze operator geeft 'false' terug indien alle expressies 'false' zijn.
!	*Logische 'niet'* Deze operator geeft 'true' indien de expressie 'false' is. Deze operator geeft 'false' indien de expressie 'true' is.

4.5 Vergelijkingsoperatoren

Vergelijkingsoperatoren geven, net zoals logische operatoren, 'true' of 'false' terug als resultaat, maar bij vergelijkingsoperatoren kunnen we ook met andere gegevenstypes werken.

Operator	Uitleg
==	Is gelijk aan
===	Is *strikt* gelijk aan
!=	Is niet gelijk aan

!==	Is niet *strikt* gelijk aan
<	Is kleiner dan
<=	Is kleiner of gelijk aan
>	Is groter dan
>=	Is groter of gelijk aan

JS-code

```
// Uitkomst is 'true' (conversie gegevenstype wordt automatisch gedaan).
10 == '10'

// Uitkomst is 'false' (gegevenstype komt niet overeen).
10 === '10'

// Uitkomst is 'true'.
'koekje' != 'Koekje'

// Uitkomst is 'true'.
10 < 50

// Uitkomst is 'true'.
10 <= 10

// Uitkomst is 'false'.
10 > 50

// Uitkomst is 'false'.
10 >= 50
```

4.6 Speciale operatoren

4.6.1 Typeof operator

De typeof operator voert geen bewerking uit, maar retourneert het gegevenstype van de variabele. Dit is handig wanneer we onze code testen.

JS-code

```
// Variabele waarde heeft als gegevenstype "boolean".
let waarde = true;
console.log("Het gegevenstype is: " + typeof waarde);
```

4.6.2 Voorwaardelijke operator

De voorwaardelijke operator kunnen we gebruiken voor voorwaardelijke expressies. Dit type expressies werkt als volgt:

- Is een voorwaardelijke expressie ‘waar’, dan worden de bijhorende statements achter het vraagteken uitgevoerd.
- Is een voorwaardelijke expressie ‘onwaar’, dan worden de bijhorende statements achter de dubbelepunt uitgevoerd.

JS-code

```
// Op zaterdag gaan we een feestje bouwen.
let dag = “Maandag”;
let uitspraak;
uitspraak = (dag == “Zaterdag”) ? “Joepie, feest!” : “Geen feest”;
console.log(uitspraak);
```