

JAVASCRIPT BASIS .....	3
1. Wat is JAVASCRIPT .....	3
2. HISTORIEK.....	3
3. WANNEER GEBRUIK JE GEEN JAVASCRIPT .....	3
4. JAVASCRIPT JARGON .....	4
5. CONSOLE (Google Chrome - Developer Tools).....	5
5.1. WINDOW Object .....	6
5.2. VARIABELEN.....	8
5.3. FUNCTIES .....	8
5.4. JAVASCRIPTOEFENINGEN - JS1 .....	9
6. VARIABELEN EN DATATYPES .....	11
7. OPERATOREN.....	14
7.1. REKENKUNDIGE OPERATOREN.....	14
7.1.1. VOORRANGSREGELS .....	15
7.1.2. SHORTHAND NOTATIES .....	15
7.2. LOGISCHE OPERATOREN.....	16
8. CONTROLESTRUCTUREN .....	17
8.1. IF - ELSE IF - ELSE STATEMENT .....	17
8.1.1. SHORTHAND NOTATIE.....	19
8.2. SWITCH.....	19
9. FUNCTIES .....	20
10. ARRAYS.....	21
11. OBJECTEN.....	23
11.1. OBJECTEN EN EIGENSCHAPPEN(PROPERTYES) .....	23
11.2. OBJECTEN EN METHODES (FUNCTIES).....	24
11.3. OBJECT MET CONSTRUCTOR .....	25
11.4. GEBRUIK VAN CONSTRUCTORS EN PROPERTYES.....	26
11.5. METHODES (FUNCTIES) TOEVOEGEN AAN EEN OBJECT .....	27
11.6. OBJECT MIXING.....	27
11.7. BUILT-IN FUNCTIES.....	27
11.8. STRING FUNCTIES .....	27
11.9. NUMBER FUNCTIES .....	29
11.10. DATUM FUNCTIES (datum object).....	30
12. ITERATIES (LOOPS OF LUSSEN) .....	31
12.1. FOR LOOP.....	31
12.2. WHILE LOOP .....	33
12.3. LOOP: CONTINUE AND BREAK.....	33
12.3.1. CONTINUE.....	33
12.3.2. BREAK .....	35
13. DOM (DOMAIN OBJECT MODEL) .....	36
13.1. GETELEMENTBYID.....	36
13.2. QUERYSELECTOR .....	37
13.3. EVENTS .....	38
13.3.1. EVENTS:VERSCHIL TUSSEN ASYNCHROON EN SYNCHROON.....	40
13.3.2. EVENTS:NAAMGEVING .....	40
13.3.2.1. ONLOAD EVENT.....	40
13.3.2.2. MOUSE EVENT.....	41
13.3.3. EVENTS:LISTENERS.....	45
14. GEBRUIK VAN JAVASCRIPT IN HTML FORMULIEREN.....	47
15. GEBRUIK VAN API'S.....	49
15.1. GOOGLE MAPS API.....	49

# JAVASCRIPT BASIS

## 1. Wat is JAVASCRIPT

Javascript IS GEEN Java. Java is een object georiënteerde programmeertaal en heeft niets te maken met Javascript. De keuze van de naamgeving was in het verleden een ongelukkig toeval.

Javascript is de programmeertaal van het web die in staat is interactiviteit te genereren zoals: click, drag, swipe, tap, ...

Javascript als programmeertaal wordt voornamelijk gebruikt bij front-end development.

Javascript kan je in elke website terugvinden. Zelfs de allergrootste sites zoals facebook en twitter gebruiken javascript. Javascript is CASE-SENSITIVE (=hoofdlettergevoelig)

Javascript wordt voornamelijk gebruikt in de front-end, nl. client-side in de browser.

Dankzij frameworks zoals bijvoorbeeld node.js kunnen we javascript ook server-side gaan gebruiken. In deze cursus zullen we het over client-side ontwikkeling hebben.

## 2. HISTORIEK

In het begin van het internet had iedere browser zijn eigen scripttaal:

- Netscape: livescript
- internet explorer: JScript

Uiteindelijk werd er overgegaan naar een universele scripttaal, nl. javascript. Javascript is ook de enige front-end taal die samen met HTML wordt gebruikt. Sedert 2009 spreken we van ECMA die de standaard is geworden, ook wel ES5 genoemd.

We spreken hier eigenlijk over ES6 (=ECMA SCRIPT 6 vanaf 2015), ES7 (=ECMA SCRIPT 7 vanaf 2016), ....

De regel die nu wordt toegepast zorgt ervoor dat elk jaar er nieuwe features worden toegevoegd. D.w.z. dat we voor 2018 reeds spreken over ES9. Dit is de standaard die nu wordt gebruikt en is nog steeds aan de basis JAVASCRIPT! We zullen hier in deze cursus ECMA ook toelichten.

OPMERKING: De huidige standaarden die gebruikt worden momenteel zijn nog steeds ES5 en ES6 omdat de features van bijvoorbeeld ES7, .... in sommige oudere browsers gewoon niet werken. Het is aan te raden om dus nog steeds de features te gebruiken van ES5 en ES6 die ruim voldoende zijn voor ons als webdevelopers.

### 3. WANNEER GEBRUIK JE GEEN JAVASCRIPT

- CSS zal zich ontfemen over oa.:
  - image swaps (rollover)
  - rollover menus
  - tooltips,...
- CSS3 transitions worden nu gebruikt i.p.v. de vroegere javascript animations.
- HTML5 form controls worden nu gebruikt i.p.v. javascript widgets

M.a.w door de evolutie in CSS3 en HTML5 zijn er veel zaken die verplaatst werden van javascript en werden opgevangen door HTML5 en CSS3. Niettegenstaande zijn er nog veel zaken die we wel dienen op te vangen met javascript en die we in deze cursus zullen overlopen.

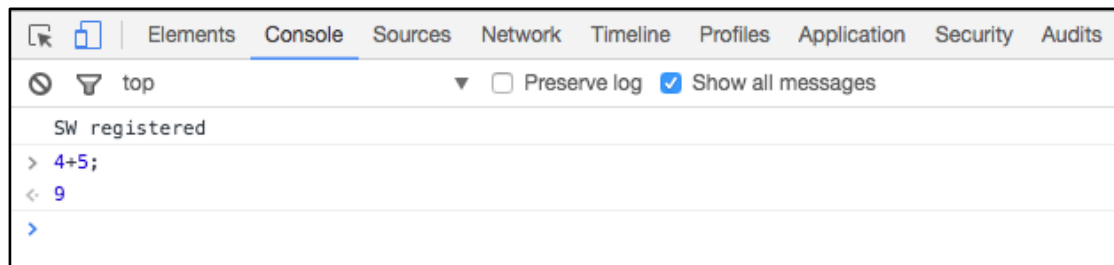
### 4. JAVASCRIPT JARGON

#### Javascript JARGON

- Data types
  - Numeriek
  - Strings
  - Booleans
  - Arrays
  - Objects
    - built-in: window, document, ... zijn allemaal objecten met hun eigen properties (eigenschappen).
- Variabelen
- Operatoren
  - + teken wordt gebruikt voor het optellen van getallen of het samenvoegen van strings
  - = teken wordt gebruikt om waarden toe te kennen aan variabelen
  - == teken wordt gebruikt om 2 waarden met elkaar te vergelijken
  - != teken is idem als het voorgaande maar dan verschillend van elkaar.
- Functies
  - zijn collecties (verzamelingen) van statements die een waarde retourneren.
- Controle structuren
  - If, else, switch
- Loops
  - For, while
- De DOM (DOCUMENT OBJECT MODEL) voor javascript
  - Veel methodes
    - Bijv. document.getElementById,...
  - Veel properties
    - value, checked, className, id,...

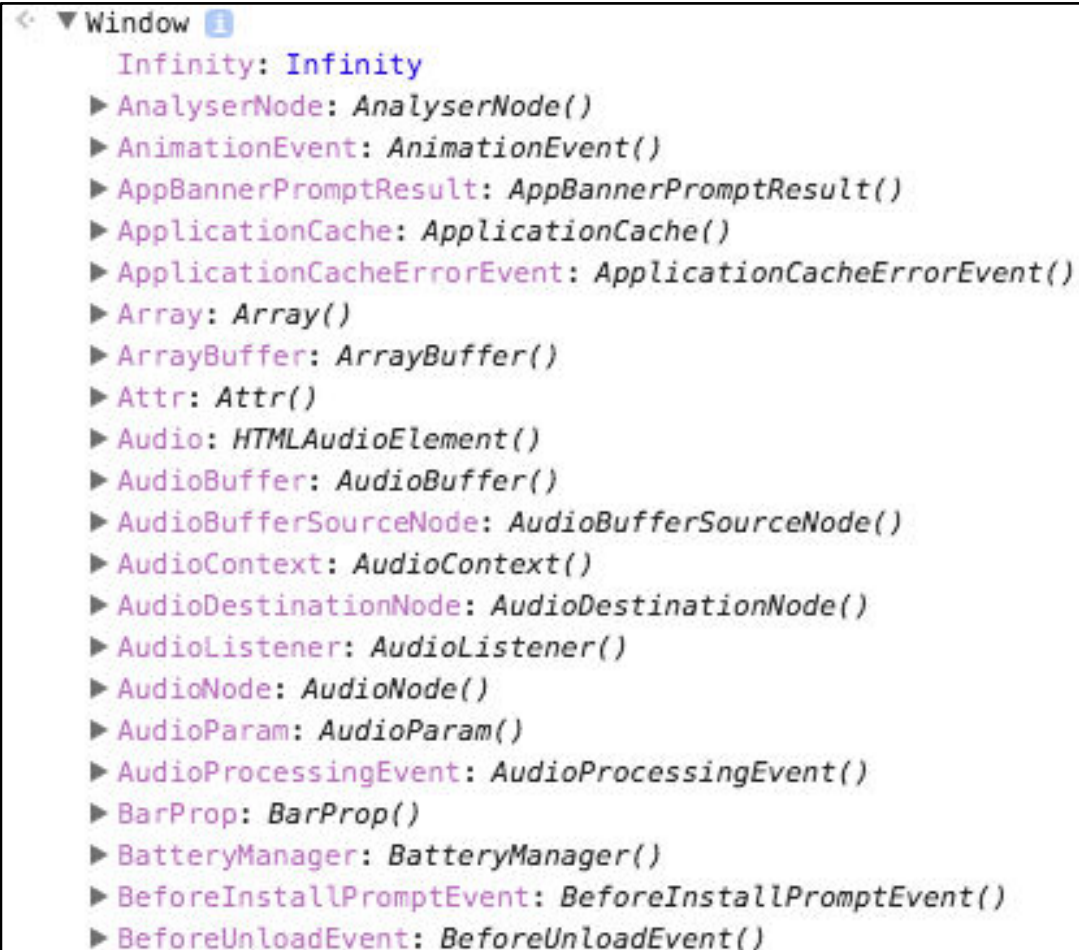
## 5. CONSOLE (Google Chrome - Developer Tools)

De Console is in staat om onmiddellijk javascript code uit te voeren. Hieronder een voorbeeld van een som binnen de console. We zullen dus eerst heel wat tijd besteden aan de console zelf, om die beter te leren kennen.



## 5.1. WINDOW Object

Javascript bestaat uit **objecten**. Het grootste object is een "**global object**" met de naam **window**. Wanneer je window bekijkt dan kan dit overweldigend overkomen omdat er zoveel mogelijkheden onder dit object zitten. Window is eigenlijk het overkoepelende venster die de mogelijkheden van javascript bevat.



```
< ▼ Window ⓘ
  Infinity: Infinity
  ▶ AnalyserNode: AnalyserNode()
  ▶ AnimationEvent: AnimationEvent()
  ▶ AppBannerPromptResult: AppBannerPromptResult()
  ▶ ApplicationCache: ApplicationCache()
  ▶ ApplicationCacheErrorEvent: ApplicationCacheErrorEvent()
  ▶ Array: Array()
  ▶ ArrayBuffer: ArrayBuffer()
  ▶ Attr: Attr()
  ▶ Audio: HTMLAudioElement()
  ▶ AudioBuffer: AudioBuffer()
  ▶ AudioBufferSourceNode: AudioBufferSourceNode()
  ▶ AudioContext: AudioContext()
  ▶ AudioDestinationNode: AudioDestinationNode()
  ▶ AudioListener: AudioListener()
  ▶ AudioNode: AudioNode()
  ▶ AudioParam: AudioParam()
  ▶ AudioProcessingEvent: AudioProcessingEvent()
  ▶ BarProp: BarProp()
  ▶ BatteryManager: BatteryManager()
  ▶ BeforeInstallPromptEvent: BeforeInstallPromptEvent()
  ▶ BeforeUnloadEvent: BeforeUnloadEvent()
```

Iedere actie in de console kan je starten met window om dan zijn eigenschappen d.m.v. een punt verder op te vragen.

```
> window.document.ATTRIBUTE_NODE
ATTRIBUTE_NODE
CDATA_SECTION_NODE
COMMENT_NODE
DOCUMENT_FRAGMENT_NODE
DOCUMENT_NODE
DOCUMENT_POSITION_CONTAINED_BY
DOCUMENT_POSITION_CONTAINS
DOCUMENT_POSITION_DISCONNECTED
DOCUMENT_POSITION_FOLLOWING
DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC
DOCUMENT_POSITION_PRECEDING
DOCUMENT_TYPE_NODE
ELEMENT_NODE
```

Je dient echter niet steeds met het window object te starten. De console herkent de onderliggende objecten ook automatisch zoals je hieronder kan zien.

```
document.ATTRIBUTE_NODE
ATTRIBUTE_NODE
CDATA_SECTION_NODE
COMMENT_NODE
DOCUMENT_FRAGMENT_NODE
DOCUMENT_NODE
DOCUMENT_POSITION_CONTAINED_BY
DOCUMENT_POSITION_CONTAINS
DOCUMENT_POSITION_DISCONNECTED
DOCUMENT_POSITION_FOLLOWING
DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC
DOCUMENT_POSITION_PRECEDING
DOCUMENT_TYPE_NODE
ELEMENT_NODE
ENTITY_NODE
```

## 5.2. VARIABELEN

Het declareren van variabelen in de console is eenvoudig. Onderstaand voorbeeld declareert de variabele x en geeft de waarde 5 mee. Telkens wanneer we nu x intikken wordt de waarde teruggegeven in de sessie van de browser. Wanneer je een browser refresh uitvoert is die waarde terug verdwenen.

```
> var x=5;
< undefined
> x
< 5
>
```

## 5.3. FUNCTIES

Ook functies kunnen worden getest in de console. Deze blijven ook tijdens de sessie bewaard. Bij een refresh van de browser verdwijnen zij net zoals variabelen. De functie alert is een built-in functie die zorgt voor een popup op je scherm.

```
< function sprekendeMuis(){
    alert('Piep piep');}
> sprekendeMuis()
```

We hebben nu kort gezien dat je in de console ook kan werken, maar je kan de console ook aansturen vanuit je webpagina zelf.

## 5.4. JAVASCRIPTOEFENINGEN - JS1

Maak een nieuwe map aan met de naam **JAVASCRIPTOEFENINGEN** en bewaar deze in een door u gekozen locatie op de pc.

Daarin maak je een map aan met de naam **js1**

Daarin maak je een eerste bestand aan met de naam: **index.html**

We gebruiken een **internal** `<script>` tag juist voor de closing body tag.

Code tussen deze `<script>` tags refereert naar javascript.

### CODE

```
index.html x
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
</head>
<body>
  <h1>Javascript Basis</h1>

  <script>
    console.log('Hallo console!');
  </script>
</body>
</html>
```

### RESULTAAT





De best practice is natuurlijk je eigen script in een **external** `<script>` file te schrijven en aan te spreken juist voor de closing body tag. Bewaar een nieuw bestand onder de map js 1: script.js

## CODE

In het script.js bestand schrijf je nu de volgende javascript code:

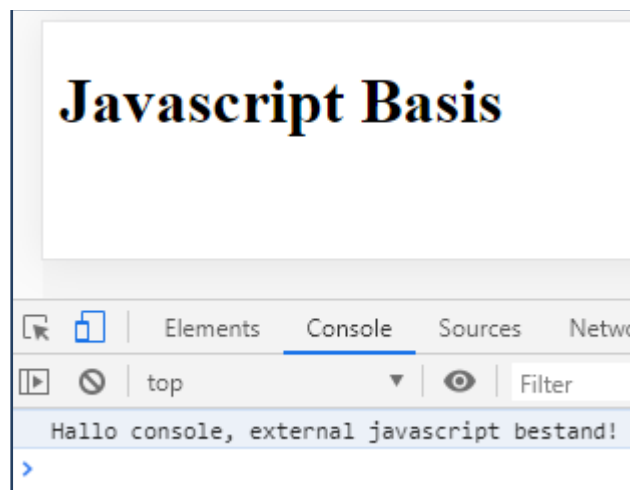
```
index.html  x  script.js  x  
console.log('Hallo console, external javascript bestand!');
```

Nu dien je het html bestand te linken met het externe script bestand. Pas daarom als volgt aan:

De script tag kent een attribuut **src** waar je de link kan toevoegen naar de locatie van het script.js bestand.

```
<body>  
  <h1>Javascript Basis</h1>  
  <script src="script.js"></script>  
</body>  
</html>
```

## RESULTAAT



## 6. VARIABELEN EN DATATYPES

We weten reeds dat Javascript case-sensitive werkt. Daarnaast dienen we ook rekening te houden met de regels voor opmaak voor variabelen. Zorg ervoor dat je variabelen een verstaanbare inhoud hebben.

Bijvoorbeeld: `a+b` vervang je beter door `getal1 + getal2`

Variabelen worden voorafgegaan door het gereserveerde woord **var**.

Een var is undefined als datatype tot er een waarde aan wordt gegeven.

EEN VAR IS PER DEFINITIE EEN STRING WANNEER ER ECHTER EEN SAMENVOEGING (CONCATENATIE) IS VAN VERSCHILLENDE DATATYPES. DIT OMDAT HET VERSCHIL NIET GEKEND IS OP DAT OGENBLIK TUSSEN BIJVOORBEELD EEN NUMBER EN EEN STRING.

Bijvoorbeeld: **var** mijnNaam

Hoe mogen variabelen NIET geschreven worden:

- GEEN gereserveerde woorden gebruiken! (var, break,...)
- NIET BEGINNEN met een cijfer
- GEEN gebruik van het koppelteken

GEEN gebruik van ongeldige tekens (&, !, ....)

De datatypes die we kunnen gebruiken zijn:

- Number: Floating point numbers (decimalen) of integers
- String: tekenreeks, tekst
- Boolean: true or false
- Undefined: Data type werd niet toegekend
- Null: 'bestaat niet'

Wanneer je gebruik wenst te maken van integers of floating-point getallen dan dien je steeds de variabelen om te zetten van een string naar deze data types.

Dit doe je met de respectievelijke **functies** `parseInt(variabele)` of `parseFloat(variabele)`.

Wanneer er niet naar een kommagetal (Float) omgezet kan worden dan krijg je een Nan foutmelding.

Nan = Not applicable number.

Open terug **script.js** onder de **map js1** en pas als volgt aan:

### CODE

```
var voorNaam = 'Tom'; //string
console.log(voorNaam);
var familieNaam = 'Vanhoutte'; //string
console.log(familieNaam);
var geboorteJaar = 1973; //number
console.log(geboorteJaar);
var functie = 'Docent'; //string
console.log(functie);
var gehuwd = true; //boolean
console.log(gehuwd);
var niets; //undefined
console.log(niets);
//toekennen van inhoud aan een undefined variabele
niets = 'niets is niet langer undefined';
console.log(niets);
/** concatenateren (samenvoegen) output als string**/
console.log(voorNaam + ' ' + familieNaam + ' is een ' + functie + ' ' + 'Full Stack Developer' + ' en is ' + gehuwd);
```

## RESULTAAT

```
Tom
Vanhoutte
1973
Docent
true
undefined
niets is niet langer undefined
Tom Vanhoutte is een Docent Full Stack Developer en is true
```

Merk op dat de laatste lijn volledig als string wordt ingelezen.

We kunnen variabelen ook laten inlezen door de gebruiker d.m.v. de **prompt** functie.

Voeg nog een laatste lijn toe en probeer maar uit in de browser:

## CODE

```
/*prompt*/
var voorNaam = prompt('Geef je voornaam in:');
console.log(voorNaam);
```

## RESULTAAT

```
Tom
Vanhoutte
1973
Docent
true
undefined
niets is niet langer undefined
Tom Vanhoutte is een Docent Full Stack Developer en is true
Tim
```

Je kan ook nagaan welk datatype wordt weergegeven. We gebruiken hiervoor de functie **typeof**.

#### CODE

```
/**TYPEOF DATATYPE**/  
console.log(typeof voorNaam);  
console.log(typeof gehuwd);
```

#### RESULTAAT

string
boolean

## 7. OPERATOREN

### 7.1. REKENKUNDIGE OPERATOREN

De basis operatoren zijn:

- deling
- vermenigvuldiging
- optelling
- aftrekking

We voegen volgende code toe aan ons JS1 bestand.

#### CODE

```
/**OPERATOREN**/  
var getal1, getal2, quotient, verschil, som, product  
getal1 = 8;  
getal2 = 4;  
  
som = getal1 + getal2; //som  
verschil = getal1 - getal2; //verschil  
product = getal1 * getal2; // vermenigvuldiging  
quotient = getal1 / getal2; //deling  
  
console.log('Som: ' + ' ' + som);  
console.log('Verschil: ' + ' ' + verschil);  
console.log('Product: ' + ' ' + product);  
console.log('Quotient: ' + ' ' + quotient);
```

#### RESULTAAT

Som: 12
Verschil: 4
Product: 32
Quotient: 2

### 7.1.1. VOORRANGSREGELS

OPMERKING: de wiskundige voorrangsregels worden toegepast in javascript.

Bijvoorbeeld: deling en aftrekking gaan voor op optelling en vermenigvuldiging.  
Haakjes gaan voor op alles in volgorde van weergave.

$5 + 5 * 5 = 30$

$(5 + 5) * 5 = 50$

#### CODE

```
/**VOORRANGSREGELS**/  
console.log(getal1 + getal2 * getal1);  
console.log((getal1 + getal2) * getal1);
```

#### RESULTAAT

40

96

### 7.1.2. SHORTHAND NOTATIES

Je kan bewerkingen ook shorthanded schrijven.

Dit is een samentrekking van het gelijk aan teken met de bewerking.

#### CODE

```
/**SHORTHAND NOTATIES**/  
x = 5  
y = 6  
  
x += y // x= x + y  
console.log('x:' + ' ' + x);
```

#### RESULTAAT

x: 11

## 7.2. LOGISCHE OPERATOREN

Bij een logische operator kunnen we 2 of meerdere variabelen met elkaar vergelijken. Het resultaat ervan zal een boolean datatype zijn, m.a.w. true or false.

Bijvoorbeeld: wanneer `getal 1 > getal 2` dan krijg je **true** als resultaat.

Hieronder heb je een overzicht van alle mogelijk logische operatoren die we kunnen gebruiken

OPERATOR	BESCHRIJVING
<code>&amp;&amp;</code>	(=AND) retourneert true als beide waarden voldoen aan de conditie Voorbeeld: <code>waarde1=0 &amp;&amp; waarde2 =0</code>
<code>  </code>	(= OR) retourneert true als één van beide waarden voldoen aan de conditie Voorbeeld: <code>waarde1=0 &amp;&amp; waarde2 =0</code>
<code>!</code>	als de operand true is wordt er false teruggegeven. Als de operand false is, wordt er true weergegeven
<code>==</code>	gelijk aan
<code>===</code>	gelijk aan en hetzelfde datatype
<code>!=</code>	verschillend van
<code>!==</code>	verschillend van en verschillend van data type
<code>&gt;</code>	groter dan
<code>&lt;</code>	kleiner dan
<code>&gt;=</code>	groter of gelijk aan
<code>&lt;=</code>	kleiner of gelijk aan

### CODE

```
/**LOGISCHE OPERATOREN**/  
var grootsteKleinste = getal1 > getal2  
console.log(grootsteKleinste);  
grootsteKleinste = getal2 > getal1 //grootsteKleinste overschreven  
console.log(grootsteKleinste);  
|  
var getal3 = '5'; //string datatype  
var getal4 = 5; //number datatype  
var gelijk = getal3 == getal4;  
console.log(gelijk);  
gelijk = getal3 === getal4; //verschillend datatype  
console.log(gelijk);
```

## RESULTAAT

Het resultaat is dus steeds true of false.

true
false
true
false

## 8. CONTROLESTRUCTUREN

### 8.1. IF - ELSE IF - ELSE STATEMENT

Met het if-else statement wordt een beslissing gevraagd binnen je code.

Voorbeeld: als **getal1 > getal2** dan voer je **code** uit. De tegenhanger is de **else**.

De **IF** staat voor het true gedeelte, de **ELSE** staat voor het false gedeelte;

Notatie in de code:

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

Hieronder vindt u 2 oefeningen

1. if en else
2. if, else if, else

## CODE

```
/**CONTROLESTRUCTUREN: IF-ELSE STATEMENT**/
var getal5 = prompt('Geef een eerste getal in');
var getal6 = prompt('Geef een tweede getal in');
if(getal5 > getal6){
    console.log(getal5 + ' is groter dan ' + getal6);
}else{
    console.log(getal5 + ' is kleiner dan ' + getal6);
}

/**CONTROLESTRUCTUREN: IF-ELSE-ELSE IF**/
var naam= prompt('Geef uw naam in:');
var beroep=prompt ('Geef uw beroep in, maak een keuze: bediende, arbeider, werkloos');
if (beroep == 'bediende'){
    console.log('Het beroep van ' + naam + ' is ' + beroep);
}else if(beroep == 'arbeider'){
    console.log('Het beroep van ' + naam + ' is ' + beroep);
}else{
    console.log('Het beroep van ' + naam + ' is ' + beroep);
}
```

## RESULTAAT



5 is kleiner dan 6
Het beroep van Tom is bediende

### 8.1.1. SHORTHAND NOTATIE

Het if-else if - else statement kan ook korter worden geschreven, dit noemen we de shorthandnotatie.

Shorthandnotatie of voorwaardelijke notatie:

**(conditie) ? waar : onwaar;**

#### CODE

```
/**SHORTHAND NOTATIE**/  
getal5 > getal6 ?  
    console.log(getal5 + ' is groter dan ' + getal6) :  
    console.log(getal5 + ' is kleiner dan ' + getal6);
```

Het resultaat is hetzelfde als de vorige oefening op het if en else statement.

### 8.2. SWITCH

Wanneer je meer dan 3 keuze mogelijkheden hebt, dan is een switch statement gemakkelijker in gebruik in vergelijking met het if - else if - else statement.

#### CODE

Het switch statement onderzoekt de waarde die een variabele bevat en voert code uit die voldoet aan die waarde. Iedere case stelt een mogelijk waarde voor, die dan ook zijn eigen code bevat. Wanneer de code wordt uitgevoerd, wordt door een break het programma beëindigd. Wanneer geen enkele case voldoet aan de variabele dan wordt default uitgevoerd.

```
/**SWITH STATEMENT**/  
var onderwijs = 'vdab';  
switch(onderwijs){  
    case 'vdab':  
        console.log('Gegeven door vdab');  
        break;  
    case 'syntra':  
        console.log('Gegeven door syntra');  
        break;  
    case 'cvo':  
        console.log('Gegeven door cvo');  
        break;  
    case 'vives':  
        console.log('Gegeven door vives');  
        break;  
    default:  
        console.log('Gegeven door een andere instelling')  
}
```

## RESULTAAT

Gegeven door vdab

## 9. FUNCTIES

Functies zijn één van de belangrijkste onderwerpen in gelijk welke programmeer- of scriptingtaal.

We hebben standaard built-in functies van javascript zelf die zeer uitgebreid zijn. Deze worden methods genoemd. Later meer hierover.

Eerst schrijven we onze eigen functie.

Notatie:

```
function naam(parameter, parameter, ....){  
    code  
}
```

## CODE

```
/**FUNCTIONS**/  
var huidigJaar = 2019;  
function berekenLeeftijd(geboorteJaar){  
    return huidigJaar - geboorteJaar;  
}  
  
var geboorteJaar = prompt('Geef uw geboortejaar in, YYYY:');  
var resultaat = berekenLeeftijd(geboorteJaar);  
  
if(resultaat >= 0){  
    console.log(console.log('Het aantal jaren tussen ' + huidigJaar + ' en ' + geboorteJaar + ' is: ' + resultaat));  
}else{  
    console.log('Uw geboorteJaar kan niet groter zijn dan het huidige jaar.');
```

## RESULTAAT

Het aantal jaren tussen 2019 en 1973 is: 46

- Enkele regel commentaar: 2 enkele slashen (//)
- /\* meerdere regels commentaar \*/

## 10. ARRAYS

Arrays in PHP is een reeks van data met waarmee we meerdere elementen van hetzelfde of verschillend gegevenstype onder één variabele kunnen opslaan. Dit bespaart ons de moeite om voor elke data een andere variabele te creëren. De arrays zijn handig om een lijst met elementen van vergelijkbare typen te maken, die toegankelijk zijn via hun index of sleutel. Stel dat we 2 namen willen opslaan en dienovereenkomstig willen afdrukken. Dit kan eenvoudig worden gedaan door het gebruik van 2 verschillende strings. Maar als in plaats van 2, willen we nu 10 variabelen gebruiken. Dan zou het erg moeilijk zijn voor de gebruiker of ontwikkelaar om zoveel verschillende variabelen te maken. Hier komt array in het spel en helpt ons om elk element binnen één enkele variabele op te slaan en maakt ook gemakkelijke toegang mogelijk met behulp van een index of een sleutel. Een array wordt gemaakt met behulp van de `array()` -functie in PHP.

Er zijn drie soorten arrays in PHP:

Geïndexeerde of numerieke arrays: een array met een numerieke index met een indexnummer startend vanaf 0.

Associatieve Arrays: een array met een indexnummer en een waarde

Meerdimensionele arrays: geneste arrays. Iedere indexnummer van de buitenste array is een array op zichzelf.

### CODE

Hier bekijken we alle facetten: aanmaken array, weergave, wijzigen, toevoegen, opzoeken/ophalen van data.

```
/**ARRAYS VULLEN**/  
var cursisten = ['Tom', 'Tim', 'Bart', 'Els']; //eerste mogelijkheid  
var cursusJaar = new Array(2017, 2018, 2019); //tweede mogelijkheid  
  
console.log(cursisten); //opsomming van de cursisten  
console.log(cursusJaar); //opsomming van de jaren  
console.log(cursusJaar.length); //aantal jaren  
console.log(cursisten.length); //aantal cursisten  
console.log(cursisten[3]); //Els wordt afgebeeld  
console.log(cursusJaar[0]); //2017 wordt afgebeeld  
  
/**WIJZIGEN ARRAY DATA**/  
cursisten[0] = 'Pieter'; //Tom wijzigt in Pieter  
console.log(cursisten);  
  
/**TOEVOEGEN VAN DATA AAN EEN ARRAY**/  
cursisten.push('Marieke'); //aan het einde van de array  
console.log(cursisten);  
cursisten.unshift('Thomas'); //aan het begin van de array  
console.log(cursisten);  
cursisten.pop(); //verwijdert de laatste cursist van de array;  
console.log(cursisten);  
cursisten.shift(); //verwijdert de eerste cursist van de array;  
  
/**ophalen van het indexnummer (locatie) van de array**/  
console.log(cursisten.indexOf('Bart')); //2 wordt afgebeeld;
```

## RESULTAAT

```
▶ (4) ["Tom", "Tim", "Bart", "Els"]
▶ (3) [2017, 2018, 2019]
3
4
Els
2017
▶ (4) ["Pieter", "Tim", "Bart", "Els"]
▶ (5) ["Pieter", "Tim", "Bart", "Els", "Marieke"]
▶ (6) ["Thomas", "Pieter", "Tim", "Bart", "Els", "Marieke"]
▶ (5) ["Thomas", "Pieter", "Tim", "Bart", "Els"]
2
```

Nog enkele veel gebruikte methodes die we kunnen toepassen bij arrays zijn:

METHODE/FUNCTIE	HOE?
<b>JOIN</b> Converteert alle elementen in een array naar een string.	<pre>var mijnArray=[10,20,30,40] console.log(mijnArray.join()); console.log(mijnArray.join('*'));</pre>
<b>SPLIT</b> Converteert een string naar een array. Dit is het omgekeerde van een join	De split functie hebben we reeds besproken in een vorig deel.
<b>REVERSE</b> Keert de volgorde van de elementen binnen een array volledig om.	<pre>var mijnArray=[10,20,30,40]; mijnArray.reverse(); console.log(mijnArray);</pre>
<b>SORT</b> Sorteert een array op alfabetische wijze van a tot z.	<pre>var mijnArray=['rood','geel','blauw']; mijnArray.Sort(); console.log(mijnArray);</pre>
<b>CONCAT</b>	2 arrays samenvoegen
<b>SLICE</b>	geeft een subarray (deel van de array) terug
<b>SPLICE</b>	voegt een element in op de aangegeven positie
<b>TOSTRING</b>	zet alle elementen in de array om naar een string

## 11. OBJECTEN

Een object lijkt een beetje op een array. In de syntax echter beginnen we een object met accolades {} ipv rechte haakjes zoals een array. Net zoals bij een array bevat deze een index en een waarde. Hier is de index steeds een veld.

### 11.1. OBJECTEN EN EIGENSCHAPPEN(PROPERTIES)

In dit onderdeel tonen we je hoe je een object maakt en de data ervan weer kan geven.

#### CODE

```
/**OBJECTEN**/  
/**OBJECTEN EN EIGENSCHAPPEN**/  
/**OBJECT MAKEN EERSTE METHODE**/  
var deelnemer= {  
  voorNaam: 'Tom',  
  familieNaam: 'Vanhoutte',  
  geboorteJaar: 1973,  
  kinderen: ['Emma', 'Elise'],  
  gehuwd: true  
};  
console.log(deelnemer);  
console.log(deelnemer.voorNaam);  
console.log(deelnemer.familieNaam);  
  
/**WIJZIGEN OBJECTEN**/  
deelnemer.voorNaam = 'Tim';  
console.log(deelnemer.voorNaam);  
  
/**OBJECT MAKEN TWEEDE METHODE**/  
var docent = new Object();  
docent.voorNaam = 'Tom';  
docent.geboorteJaar = 1973;  
  
console.log(docent);
```

## RESULTAAT

In dit resultaat zie je duidelijk het verschil met een array. Een object geeft je veld (index) en waarde (value) terug voor ieder item. Een object op zich is dus eigenlijk ook een array (reeks)

```
▶ {voorNaam: "Tom", familieNaam: "Vanhoutte", geboorteJaar: 1973, kinderen: Array(2), gehuwd: true}
Tom
Vanhoutte
Tim
▶ {voorNaam: "Tom", geboorteJaar: 1973}
```

## 11.2. OBJECTEN EN METHODES (FUNCTIES)

Voor een object kan je zelf functies schrijven. In ons voorbeeld berekenen we de leeftijd van een deelnemer met een zelf geschreven methode die enkel tot dit object behoort.

## CODE

In het object voegen we een functie toe die je geboortejaar aftrekt van het huidige Jaartal tijdens schrijven van deze cursus.

Hier maken we ook voor het eerst gebruik van het **this** keyword.

Wanneer het keyword this wordt gebruikt dan slaat dit steeds standaard op het window.object van de browser. Maar in dit geval gaat this over het item in code waar hij in staat, dus m.a.w. het object deelnemer.

```

/**OBJECTEN**/
/**OBJECTEN EN EIGENSCHAPPEN**/
/**OBJECT MAKEN EERSTE METHODE**/
var deelnemer= {
  voorNaam: 'Tom',
  familieNaam: 'Vanhoutte',
  geboorteJaar: 1973,
  kinderen: ['Emma', 'Elise'],
  gehuwd: true,
  berekenLeeftijd: function(){
    this.leeftijd = 2019 - this.geboorteJaar;
  }
};
console.log(deelnemer);
console.log(deelnemer.voorNaam);
console.log(deelnemer.familieNaam);

/**WIJZIGEN OBJECTEN**/
deelnemer.voorNaam = 'Tim';
console.log(deelnemer.voorNaam);

/**OBJECT MAKEN TWEEDE METHODE**/
var docent = new Object();
docent.voorNaam = 'Tom';
docent.geboorteJaar = 1973;

console.log(docent);

/**OBJECT METHODE AANSPREKEN**/
deelnemer.berekenLeeftijd(); //berekent de leeftijd
console.log(deelnemer); //toont het volledige object met leeftijd

```

## RESULTAAT

```

▼ {voorNaam: "Tim", familieNaam: "Vanhoutte", geboorteJaar: 1973, kinderen: Array(2), gehuwd: true, ...} ⓘ
  ► berekenLeeftijd: f ()
    familieNaam: "Vanhoutte"
    geboorteJaar: 1973
    gehuwd: true
  ► kinderen: (2) ["Emma", "Elise"]
    leeftijd: 46
    voorNaam: "Tim"
  ► __proto__: Object

```

### 11.3. OBJECT MET CONSTRUCTOR

Tot nu toe hebben we telkens 1 object kunnen vullen, hierdoor zijn we dus gelimiteerd.

Stel je voor dat je 1 object kan creëren die je oneindig kan gebruiken voor alle merken, types en kleuren van auto's.

Hiervoor gebruiken we een OBJECT CONSTRUCTOR.

```

function auto(merk, type, kleur){
  this.merk = merkNaam;
  this.type = typeNaam;
  this.kleur= typeKleur;
}

```



In javascript wordt **this** aangeroepen. In dit geval is **this** de eigenaar van het object, aangezien we in een object werken.  
Wanneer je het gereserveerde woord **this** in andere delen van code tegenkomt, is **this** altijd de eigenaar van dat deel van de code.

Van bovenstaande constructor kunnen we MEERDERE INSTANTIES VAN EEN OBJECT maken.

```
var auto1 = new auto('Mercedes', 'C-klasse', 'Rood');  
var auto2 = new auto('Mercedes', 'E-klasse', 'Rood');  
var auto3 = new auto('BMW', 'Z4', 'Blauw');
```

#### 11.4. GEBRUIK VAN CONSTRUCTORS EN PROPERTIES

Voorbeeld1: object variabelen op het scherm weergeven.

`auto1.merk + " , " + auto1.type + " , " + auto1.kleur; //meest gebruikt!`

**of**

`auto1["merk"] + " , " + auto1["type"] + auto1["kleur"];`

### 11.5. METHODES (FUNCTIES) TOEVOEGEN AAN EEN OBJECT

Het toevoegen van de methodes (functies) doet men BINNEN de constructor onder de eigenschappen

CONSTRUCTOR:

```
function auto(merk, type, kleur){
    this.merk = merkNaam;
    this.type = typeNaam;
    this.kleur= typeKleur;
    this.wijzigKleur = function(anderkleur){
        this.kleur = anderkleur;
    }
}
```

GEBRUIK:

```
var auto1 = new auto('Mercedes', 'C-klasse', 'Rood');
auto1.wijzigkleur('Blauw');
```

```
document.getElementById("divResultaat").innerHTML =
"Het kleur van mijn wagen is:" + auto1.kleur;
```

### 11.6. OBJECT MIXING

Het is mogelijk om objecten te mixen met andere objecten zoals bijvoorbeeld een Array. Hieronder zie je een array die gevuld is met objecten:

```
var autos=[
    {Merk:'Mercedes',Type:'C-klasse',Kleur:'rood'},
    {Merk:'Mercedes',Type:'E-klasse',Kleur:blauw},
    {Merk:'BMW',Type:'Z-4,Kleur:'geel'}
];
```

### 11.7. BUILT-IN FUNCTIES

Op de developer site van mozilla kan je de uitleg vinden van alle mogelijke functies die javascript kan bieden. Hieronder volgt een overzicht van de meest gebruikte functies!

### 11.8. STRING FUNCTIES

HOE?	FUNCTIE	GEBRUIK
HOOFDLETTERS	toUpperCase()	mijnVariabele.toUpperCase()
KLEINE LETTERS	toLowerCase()	mijnVariabele.toLowerCase()
STRINGS SAMENVOEGEN	concat()	mijnVariabele.concat(string1,string2)
KARAKTER UIT EEN STRING WEERGEVEN	charAt(0) 0 = positie teken	mijnVariabele.charAt(0). Geeft het 0de teken van een string weer
GEEF HET DECIMAAL GETAL UIT ASCII	charCodeAt(0)	mijnVariabele.charCodeAt(0). = 72

<b>STRING NAAR EEN ARRAY CONVERTEREN</b>	<code>split();</code>	<code>txt.split(",")</code> . Splits een string in een element van een array waar je een komma vindt.
<b>SUBSTRING</b>	<code>substring()</code>	<code>mijnVariabele.substring(0, 3)</code> . Start bij het eerste karakter en toon 3 karakters van de string.
<b>GEEF DE POSITIE VAN EEN KARAKTER UIT EEN STRING</b>	<code>IndexOf()</code>	<code>mijnVariabele = "webontwikkelaars"</code> <code>mijnVariabele.IndexOf('o') =&gt; 4</code> <code>mijnVariabele.IndexOf('u') =&gt; -1 =&gt;</code> komt niet voor in de variabele!
<b>LAATSTE POSITIE WAAR EEN KARAKTER VOORKOMT</b>	<code>LastIndexOf()</code>	<code>mijnVariabele = "webontwikkelaars"</code> <code>mijnVariabele.LastIndexOf('w') =&gt; 7</code>
<b>VERVANGEN KARAKTER</b>	<code>Replace()</code>	<code>mijnVariabele = "webontwikkelaars"</code> <code>mijnVariabele.Replace('e', 'a') =&gt;</code> wabontwikkelaars

## 11.9. NUMBER FUNCTIES

HOE?	FUNCTIE	GEBRUIK
GETAL NAAR STRING	toString()	mijnVariabele.toString()
AFRONDING NA DE KOMMA MET EXPONENT	toExponential()	mijnVariabele.toExponential(2). Afronding tot 2 cijfers na de komma met Exponent.
AFRONDING NA DE KOMMA ZONDER EXPONENT	toFixed()	mijnVariabele.toFixed() toFixed(2) wordt gebruikt om geld weer te geven.
EEN NUMMER ALS NUMMER WEERGEVEN ZONDER PARSEINT OF PARSEFLOAT	valueOf()	mijnVariabele.valueOf(). (50+50).valueOf() = 100
CONVERTEER VARIABLES NAAR NUMMERS	Number()	Number(mijnVariabele) Zet alles om naar nummers. mijnVariabele = true => 1 mijnVariabele = false => 0 mijnVariabele = new Date() => 120305.... mijnVariabele = "50" => 50
STRING NAAR EEN ARRAY CONVERTEREN	split();	txt.split(","). Splits een string in een element van een array waar je een komma vindt.
CONVERTEREN NAAR EEN VOLLEDIG NUMMER	parseInt()	parseInt(mijnVariabele) parseInt("50"); => 10 parseInt("50.8") => 50 parseInt("50 uren") => 50
CONVERTEREN NAAR NUMMER OF KOMMAGETAL	parseFloat()	parseFloat(mijnVariabele) parseFloat("50.8") => 50.8 parseFloat("50") => 50 parseFloat("50 uren") => 50
MATH		
GETAL PI	Math.PI	
AFRONDEN	Math.round()	0.6 => 1 0.5 => 1 0.4 => 0
MACHTSVERHEFFING	Math.pow(4,2)	4 tot de 2de = 16
VIERKANTSWORTEL	Math.sqrt(16)	vierkantswortel van 16 = 4
ABSOLUTE WAARDE	Math.abs()	Geeft de absolute positieve waarde van een getal
AFRONDEN NAAR BOVEN	Math.ceil()	Math.ceil(3.2) => 4
AFRONDEN NAAR	Math.floor	Math.floor(3.2) => 3

<b>BENEDEN</b>		
<b>SINUS</b>	Math.sin()	Math.sin(90 * Math.PI/180) => 1 => sinus van 90 graden
<b>COSINUS</b>	Math.cos()	Math.cos(0 * Math.PI/180) => 1 => cos van 0 graden
<b>MINIMUM VAN EEN REEKS</b>	Math.min()	Math.min(0,-3,5,8) => -3
<b>MAXIMUM VAN EEN REEKS</b>	Math.max()	Math.max(0,8,9,10) => 10
<b>RANDOM NUMMER</b>	Math.random()	geeft een random nummer weer

### 11.10. DATUM FUNCTIES (datum object)

De datum functie is eigenlijk een OBJECT type. Objecten hebben de eigenschap om meerdere parameters te gaan gebruiken. Een object met zijn eigen parameters noemen we een constructor. Een constructor kan meerdere instanties van een object genereren.

Om een object van het type DATE aan te spreken dienen we gebruik te maken van het gereserveerde woord **NEW**.

```
new Date()
new Date(milliseconds)
new Date (dateString)
new Date(jaar, maand, dag, uren, minuten, seconden, milliseconden)
```

## 12. ITERATIES (LOOPS OF LUSSEN)

JavaScript-lussen worden gebruikt om een codeblok herhaaldelijk uit te voeren, totdat aan een bepaalde voorwaarde is voldaan. JavaScript biedt verschillende opties om herhaaldelijk een codeblok uit te voeren, inclusief while, do while, for en for-in.

### 12.1. FOR LOOP

De meest gebruikte loop in javascript is de for loop.  
Hieronder zie je drie voorbeelden van for loops.

Notatie:

```
for(teller;conditie;wijziging){  
    uitvoering code  
}
```

teller = startpunt van de lus

conditie = de voorwaarde of het aantal keren dat de code dient te worden uitgevoerd.

wijziging = staat voor de wijziging van de teller.

#### CODE

In ons eerste voorbeeld beginnen we als startpunt met een teller die de naam i draagt en start op 1. De conditie bepaalt dat de lus zolang wordt uitgevoerd tot de teller = 10.

i++ telt telkens 1 op bij de teller.

```
/**FOR LOOP**/  
for(var i = 1; i <= 10; i++){  
    console.log(i);  
}  
  
for(var i = 1; i <= 10; i+=2){  
    console.log(i);  
}  
  
for(var i = 10; i > 1; i--){  
    console.log(i);  
}
```

## RESULTAAT

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

De for-loop kan gebruikt worden om bijvoorbeeld een **array uit te lezen**.

## CODE

Hieronder zie je dat het conditie gedeelte wordt bepaald d.m.v. de length methode. Herinner je dat deze functie het aantal elementen van een array kan weergeven. Ieder afzonder element uit de array wordt vervolgens weergegeven door de teller [i] te gebruiken (**cursisten[i]**).

```
/**FOR LOOP: ARRAY**/  
var cursisten = ['Tom', 'Tim', 'Bart','Els'];  
for(var i=1;i < cursisten.length; i++){  
    console.log(cursisten[i]);  
}
```

## RESULTAAT

Merk op dat **Tom** hier niet wordt weergegeven omdat we niet van het nulde element tellen maar van het eerste. (**var i = 1**)

Tim
Bart
Els

## 12.2. WHILE LOOP

In principe kan je met iedere soort van loop elke code schrijven. De while loop wordt als volgt gebruikt.

Notatie:

```
while(conditie){
    code
}
```

While loop = ZOLANG aan een conditie wordt voldaan DOE de code.

### CODE

```
/**WHILE LOOP: ARRAY**/
var i = 0
while(i < cursisten.length){
    console.log(cursisten[i]);
    i++;
}
```

### RESULTAAT

Tom
Tim
Bart
Els

## 12.3. LOOP: CONTINUE AND BREAK

### 12.3.1. CONTINUE

Wanneer aan een conditie wordt voldaan, wordt de continue gebruikt om deze NIET uit te voeren, maar verder te gaan met het volgende item in bijvoorbeeld een array. Zie voorbeeld hieronder

### CODE

```
/**LOOP:CONTINUE**/
var data = ['Tim', 'Tom', 1980, 1973, 'designer', 'developer'];
for(var i = 0; i < data.length; i++){
    if(typeof data[i] === 'string') continue; //wanneer gelijk aan string DOE NIKS!
    console.log(data[i]); // wanneer verschillend van string, druk af!
}
```

### RESULTAAT



1980
------

1973
------

### 12.3.2. BREAK

Een break is een HARDE onderbreking van de code wanneer aan een conditie wordt voldaan.

#### CODE

Hieronder onderbreken we het lopen van de array wanneer hij een item tegenkomt die GEEN STRING is. In dit geval is dit het jaartal 1980. De code wordt onderbroken en de andere items worden niet meer gecontroleerd!

```
/**LOOP: BREAK**/  
var data = ['Tim', 'Tom', 1980, 1973, 'designer', 'developer'];  
for(var i = 0; i < data.length; i++){  
    if(typeof data[i] !== 'string') break; //springt uit de array wanneer verschillend van een string  
    console.log(data[i]); // wanneer verschillend van string, druk af!  
}
```

#### RESULTAAT

Tim
Tom

### 13. DOM (DOMAIN OBJECT MODEL)

Tot nu toe hebben we gebruikt gemaakt van de console in de webbrowser om onze resultaten weer te geven. De console zal steeds een belangrijk item zijn in javascript om vlug waarden te kunnen raadplegen, maar eigenlijk dienen deze waarden op een webpagina zelf te worden weergegeven. Een webpagina wordt opgebouwd binnen de DOM (= DOMAIN OBJECT MODEL).

De DOM is een benadering van gestructureerde elementen. Dit kan gaan over HTML, XML, JSON, XHTML.

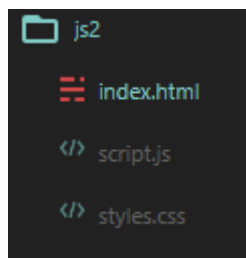
JAVASCRIPT is in staat om te communiceren met de DOM.

#### 13.1. GETELEMENTBYID

Hier zullen we de interactie tonen tussen een HTML element en ons script bestand.

Maak een map JS2 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.

De bedoeling is dat we de script code toe te kennen aan een html element binnen de DOM. In dit voorbeeld zal dit de div met id = resultaat zijn. GetElementById zorgt voor de verbinding tussen de div tag resultaat in HTML en ons script bestand.



#### HTML

```
<!DOCTYPE html>
<html Lang="nl">

<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <div id="resultaat"> </div>
  <script src="script.js"></script>
</body>

</html>
```

#### SCRIPT

```
var naam = prompt('Vul uw naam in:', 'uw naam');
var weergave = 'De naam die u invoerde was:' + naam;
document.getElementById('resultaat').innerHTML = weergave;
```

RESULTAAT

# Javascript Basis

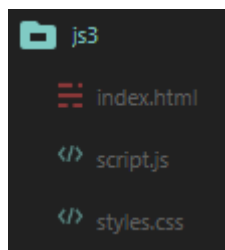
## DOM

De naam die u invoerde was:Tom

### 13.2. QUERYSELECTOR

Mijn persoonlijke favoriet is de queryselector omdat hij zowel id als class kan aanspreken.

Maak een map JS3 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.



### HTML

```
<!DOCTYPE html>
<html lang="nl">

<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <div id="resultaat"> </div>
  <div class="basis"></div>

  <div class="dynamisch1"></div>
  <div class="dynamisch2"></div>
  <div class="dynamisch3"></div>
  <script src="script.js"></script>
</body>
</html>
```

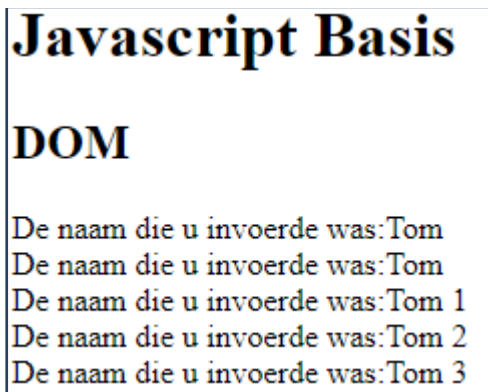
## SCRIPT

Hieronder ziet u dat zowel de class als id kunnen aangesproken worden met de queryselector. Hierboven hebben we tevens 3 verschillende classes bijgemaakt met de beginvariabele **dynamisch** telkens met een ander cijfer. We gebruiken een for loop om deze 3 classes in één keer te injecteren van ons resultaat.

```
var naam = prompt('Vul uw naam in:', 'uw naam');
var weergave = 'De naam die u invoerde was:' + naam;
document.querySelector('#resultaat').textContent = weergave;
document.querySelector('.basis').textContent = weergave;

for(i=1; i <= 3; i++){
    document.querySelector('.dynamisch' + i ).textContent = weergave + ' ' + i;
}
```

## RESULTAAT

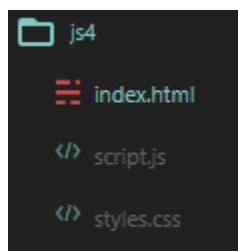


### 13.3. EVENTS

Alhoewel javascript geen volledig object georiënteerde taal is (OOP= Object Oriented Programming Language), kunnen we toch gebruiken maken van events. Een event is een gebeurtenis. Een klik op een button in de pagina is dus per definitie een event (=gebeurtenis).

Meerdere events (bijvoorbeeld knoppen) kunnen door een gebruiker op de pagina aangeklikt worden. Javascript kan de uitvoering van deze events asynchroon behandelen.

Maak een map JS4 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.



## HTML

```
<!DOCTYPE html>
<html lang="nl">

<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <button id="btnStart" onClick="Naam()">Klik op mij</button>
  <div id="resultaat"> </div>
  <script src="script.js"></script>
</body>

</html>
```

## SCRIPT

```
function Naam(){
  var naam = prompt('Vul uw naam in:', 'uw naam');
  var weergave = 'De naam die u invoerde was:' + naam;
  document.querySelector('#resultaat').textContent = weergave;
}
```

## CSS

Dit is geen cursus HTML of CSS, maar we kunnen dus ook de styling van onze elementen aanpassen indien gewenst.

```
#btnStart{
  background: red;
  color: white;
}
```

## RESULTAAT



### 13.3.1. EVENTS:VERSCHIL TUSSEN ASYNCHROON EN SYNCHROON.

Javascript is in staat om events Asynchroon te verwerken. Simpel uitgelegd wil dit zeggen dat tegelijkertijd meerdere events naast elkaar kunnen worden uitgevoerd. Wanneer een programma synchroon werkt, wil dit zeggen dat een event pas kan uitgevoerd worden wanneer het vorige event werd afgewerkt. In de huidige programmeerwereld is asynchroon programming een must!

### 13.3.2. EVENTS:NAAMGEVING

Wanneer we ons voorbeeld nemen van een button in een html pagina, dan zullen we het **onclick()** event aanroepen om onze code uit te voeren. Hier spreken we dan ook van **event handlers**.

Andere voorbeelden van veel gebruikte events zijn:

- onload
- onchange
- onclick
- onmouseover
- onkeyup
- onfocus
- onkeydown
- onmouseout

#### 13.3.2.1. ONLOAD EVENT

Maak een map JS5 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.

Het onload event wordt pas uitgevoerd wanneer een pagina volledig is geladen!

#### HTML

```
<!DOCTYPE html>
<html Lang="nl">

<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <h3>Events</h3>
  <div id="resultaat"> </div>
  <script src="script.js"></script>
</body>
</html>
```

## SCRIPT

```
window.onload = function(){  
    document.querySelector('#resultaat').textContent = 'Het document is geladen';  
}
```

## RESULTAAT



### 13.3.2.2. MOUSE EVENT

Maak een map JS6 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.

Er kunnen verschillende javascript events getriggerd worden wanneer je bijvoorbeeld over een html element heen zou bewegen, klikken, ...

Hieronder enkele mogelijk voorbeelden:

## HTML

```
<!DOCTYPE html>  
<html Lang="nl">  
  
  <head>  
    <meta charset="UTF-8">  
    <title>Javascript Basis</title>  
    <link rel="stylesheet" href="styles.css">  
  </head>  
  
  <body>  
    <h1>Javascript Basis</h1>  
    <h2>DOM</h2>  
    <h3>Events</h3>  
    <h4>Mouse events</h4>  
    <h5>Mouse over en mouse out</h5>  
    <div id="binnen"></div>  
    <div id="buiten"></div>  
    <script src="script.js"></script>  
  </body>  
</html>
```



## CSS

```
#binnen{  
  height: 200px;  
  width: 200px;  
  border: 2 px solid black;  
  background: yellow;  
}
```

## SCRIPT

De variabele **binnen** stelt het vierkant voor. Elk event hieronder spreekt voor zich. Probeer één voor één uit en bekijk het resultaat in je browser.

```
window.onload = function(){  
  var binnen = document.getElementById('binnen');  
  var buiten = document.getElementById('buiten');  
  var x = 0;  
  var z = 0  
  
  binnen.onmousemove = function(){ //bewegen van de muis  
    binnen.innerHTML = x+=1;  
  }  
  binnen.onmouseover = function(){ //bewegen van de muis over een html element  
    binnen.style.backgroundColor = '#FAC';  
  }  
  binnen.onmouseout = function(){ //verlaten van de muis van een html element  
    buiten.innerHTML += 'De muis beweegt uit het vierkant<br>';  
    binnen.style.backgroundColor = '#FFF';  
  }  
  binnen.onmousedown = function(){ //Linkermuis ingedrukt  
    binnen.innerHTML += z+=1;  
  }  
  binnen.onmouseleave = function(){ //verlaten html element  
    binnen.innerHTML += z-=1;  
  }  
}
```

## RESULTAAT

# Javascript Basis

## DOM

### Events

#### Mouse events

##### Mouse over en mouse out

810

De muis beweegt uit het vierkant

Er zijn heel wat events in javascript.  
Hieronder de volledige gekende lijst:

abort	hashchange	resize
afterprint	input	reset
animationend	invalid	scroll
animationiteration	keydown	search
animationstart	keypress	seeked
beforeprint	keyup	seeking
beforeunload	load	select
blur	loadeddata	show
canplay	loadedmetadata	stalled
canplaythrough	loadstart	storage
change	message	submit
click	mousedown	suspend
contextmenu	mouseenter	timeupdate
copy	mouseleave	toggle
cut	mousemove	touchcancel
dblclick	mouseover	touchend
drag	mouseout	touchmove
dragend	mouseup	touchstart
dragenter	mousewheel	transitionend
dragleave	offline	unload
dragover	online	volumechange
dragstart	open	waiting
drop	pagehide	wheel
durationchange	pageshow	
ended	paste	
error	pause	
focus	play	
focusin	playing	

focusout	popstate
fullscreenchange	progress
fullscreenerror	ratechange

### 13.3.3. EVENTS:LISTENERS

Maak een map JS7 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.

Een andere schrijfwijze voor het schrijven van een event handler zijn event 'listeners'.

Voorbeeld: mouse event met event listener

#### HTML

```
<!DOCTYPE html>
<html Lang="nl">

<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <h3>Events</h3>
  <h4>Mouse events</h4>
  <h5>Mouse click event</h5>
  <button id="btnStart">Hoeveel muisklikken?</button>
  <div id="resultaat"></div>
  <script src="script.js"></script>
</body>

</html>
```

#### SCRIPT

```
var numClicks = 0 //globale variable
window.onload = function(){
  var resultaat = document.getElementById('resultaat');
  document.getElementById('btnStart').addEventListener('click',function(){
    numClicks++;
    resultaat.innerHTML = numClicks + ' keer geklikt.';
  }, false);
}
```

## RESULTAAT

### Javascript Basis

#### DOM

#### Events

#### Mouse events

#### Mouse click event

Hoeveel muisklikken?

9 keer geklikt.

## 14. GEBRUIK VAN JAVASCRIPT IN HTML FORMULIEREN

Maak een map JS8 aan telkens met een index.html bestand, styles.css bestand en een script.js bestand erin.

In formulieren gebruiken we o.a. tekstvelden om waarden in te vullen. In HTML hebben jullie reeds gezien dat dit over input velden gaat. De button zal de tekst die wordt ingevuld in het tekstveld ophalen en via javascript weergeven in het resultaat.

### HTML

```
<head>
  <meta charset="UTF-8">
  <title>Javascript Basis</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <h1>Javascript Basis</h1>
  <h2>DOM</h2>
  <h3>Events</h3>
  <h4>Form events</h4>
  <input type="text" id="textNaam" placeholder="uw naam...">
  <button id="btnStart">Toon boodschap</button>
  <div id="resultaat"></div>
  <script src="script.js"></script>
</body>
</html>
```

### SCRIPT

```
var naam = ''; //globale variable
window.onload = function(){
  var resultaat = document.getElementById('resultaat');
  //event handler voor de button
  document.getElementById('btnStart').addEventListener('click',function(){
    naam= document.getElementById('textNaam').value;
    resultaat.innerHTML = '<h5>Welkom ' + naam + '</h5>';
  }, false);
}
```

### RESULTAAT

# Javascript Basis

## DOM

### Events

#### Form events

Welkom Tom

## 15. GEBRUIK VAN API'S

API = Application programming interface.

Veel sites stellen hun API ten dienste van iedereen. Eigenlijk is dit gebruik maken van hun broncode binnen onze sites.

Voorbeelden van API's zijn:

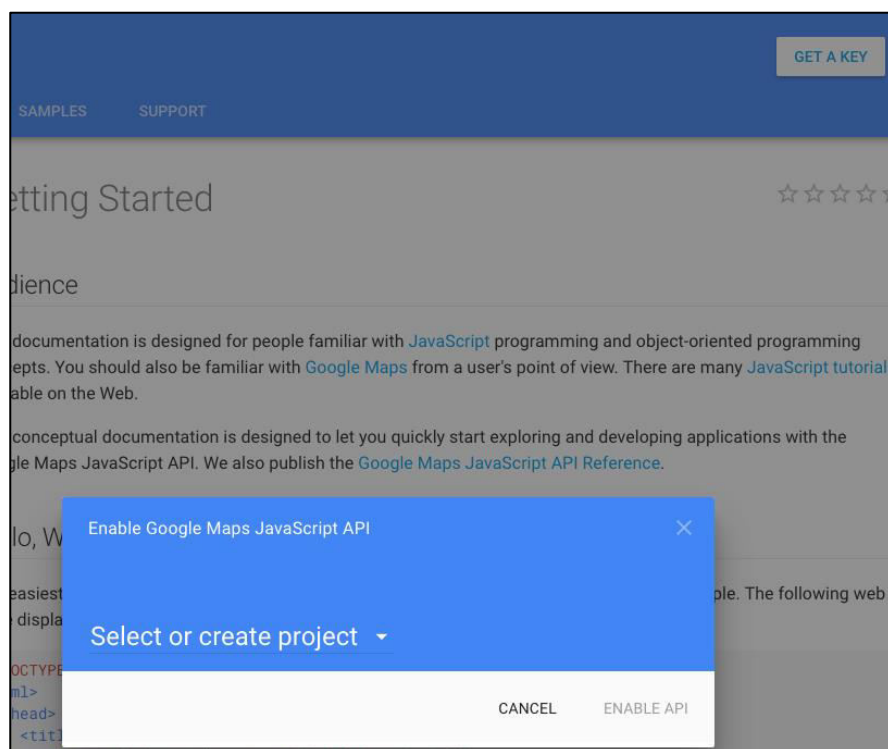
- Google Maps
- Bing Maps
- Youtube
- Vimeo
- Soundcloud
- ...

### 15.1. GOOGLE MAPS API

Wij zullen de API van google maps aanspreken in onze contact.html pagina. De documentatie voor de google maps API bevindt zich op de volgende link:

<https://developers.google.com/maps/documentation/javascript>

Het eerste wat we moeten doen op de API te gebruiken is een projectnaam opgeven en een KEY genereren.





In de contactpagina voeg je een div met de naam map toe met een hoogte van 500px. Deze div zal de google maps pagina bevatten.

```
#map {  
  height: 500px;  
}
```

Vergeet ook de div niet binnen je html pagina te plaatsen op de locatie waar de kaart dient te verschijnen.

```
<div id="map"></div>
```

In de javascript tags kopiëren we de volledige script en plakken die in onze pagina. Wanneer je de pagina zal herladen is er reeds een kaart zichtbaar.

```
<script>  
  var map;  
  function initMap() {  
    map = new google.maps.Map(document.getElementById('map'), {  
      center: {lat: -34.397, lng: 150.644},  
      zoom: 8  
    });  
  }  
</script>  
<script src="https://maps.googleapis.  
  com/maps/api/js?key=AIzaSyCLHqifZHNuBLa3uaMVihayqTHQt8njNbQ&callback=initMap"  
  async defer></script>  
</body>
```

De volgende cursus is een uitbreiding van javascript handelt over ECMA SCRIPT