



Webdesign Javascript

Funcities

Functie

```
function myFunction(param1, param2){  
    //Do a lot of things  
    //Hopefully interesting things  
  
    /* optioneel */  
    return param;  
}  
var aap = myFunction("iets", "ietsanders");
```

- Om stukken code te bundelen
- Eventueel met return statement
- Geen of meerdere invoerparameters
- Geen ; bij functiedefinitie, wel ; bij functieaanroep



Webdesign Javascript

Funcities

Functie parameters

```
var mijnNaam="Willems";  
function printNaam(naam){  
    document.getElementById("divResult").innerHTML =  
        "<h2>Uw naam is: " + naam + "</h2>";  
}  
printNaam(mijnNaam);
```

- De naam van de variabele die meegegeven wordt `mijnNaam` moet niet gelijk zijn aan de naam van de parameter `naam`
- De parameter `naam` krijgt de inhoud van de variabele `mijnNaam`
- Binnen de functie is `mijnNaam` bekend als `naam` (call by value)
- De waarde van `naam` wordt bij elke functieaanroep ververs en blijft niet bewaard
- Alle type variabele mogen gebruikt worden als parameter, ook weer andere functies.



Webdesign Javascript

Scoping

Scoping

- De scope is de plaats waar je op dat moment toegang hebt tot variabelen, functies of objecten

```
// Globale scope van de website
// hier kan ik niet aan naamDocent
// => alert(naamDocent) geeft undefined

function myFunc(){
    var naamDocent = "Willems";

    // Lokale scope van de functie myFunc()
    // hier kan ik wel aan naamDocent
    // => alert(naamDocent) geeft "Willems"
};
```



Webdesign Javascript

Scoping

Scoping

- De scope is de plaats waar je op dat moment toegang hebt tot variabelen, functies of objecten
- Globale scope
 - = Alles wat buiten de functies is gedeclareerd
 - + alles wat zonder 'var' is gedeclareerd
- Lokale scope
 - = Scope binnen de functie
 - = Kan enkel daar aangesproken worden



Webdesign Javascript

Scoping

Scoping

```
function first(){
    var a = "alle ";
    function second(){
        var b = "eendjes ";
        function third(){
            var c = a + b + "zwemmen in het water!";
            alert(c); //alle eendjes zwemmen in het water
        }
        alert(c); //undefined
        third();
    }
    alert(b); //undefined
    second();
}
alert(a); //undefined
first(); //undefined undefined alle eendjes zwemmen in het water
```

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

5



Webdesign Javascript

Scoping

Let versus var

- Sinds ES6 kan je ook gebruik maken van `let` ipv `var`.
- Met `let` definieer je specifiek lokale variabelen. De variabele is pas gekend nadat deze werd aangeroepen:

```
var x = 4;
document.write(x, "\n"); // var x aanroepen na definitie resulteert in 4

let y = 7;
document.write(y, "\n"); // let y aanroepen na definitie resulteert in 7

document.write(z, "\n"); // var z aanroepen voor definitie resulteert
var z = 5;                // in undefined

document.write(a);        // let a aanroepen voor definitie resulteert
let a = 3;                // in een error vb. in Chrome:
```

✖ Uncaught ReferenceError: a is not defined

Bron: <https://www.geeksforgeeks.org/difference-between-var-and-let-in-javascript/>

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

6



Webdesign Javascript

Scoping

Let versus var

- De lokale variabelen gedefinieerd met `let` is enkel gekend binnen een blok:

```
for(var i=0; i<5; i++){  
    console.log(i); // i bestaat, dus 0,1,2,3 en 4  
}  
// worden gelogd in de console  
console.log(i);    // i bestaat nog steeds, dus 5  
// wordt gelogd in de console  
  
for(let j=0; j<5; j++){  
    console.log(j); // j bestaat binnen de blok, dus 0,1,2,3 en 4  
}  
// worden gelogd in de console.  
console.log(j);    // buiten de blok bestaat j niet meer. Dit  
// resulteert in een foutmelding. Vb. In Chrome:
```

❌ ▶ Uncaught ReferenceError: j is not defined

Bron: <https://codeburst.io/difference-between-let-and-var-in-javascript-537410b2d707>

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

7



Webdesign Javascript

Scoping

Let versus var

- Het herdefiniëren van de lokale variabelen met `let` is niet mogelijk:

```
var a = 5;  
var a = 7; // var a kan eenvoudig opnieuw worden gedeclareerd  
  
let b = 5;  
let b = 7; // let b kan niet opnieuw worden gedeclareerd.  
// Dit resulteert in een error. Vb. In Chrome:
```

❌ Uncaught SyntaxError: Identifier 'b' has already been declared

Bron: <https://codeburst.io/difference-between-let-and-var-in-javascript-537410b2d707>

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

8

**Meerdere waarden retourneren**

```
function kwadraat(getal){
    var kwadraat = getal * getal;
    var resultaat = "OK";
    return {
        uitkomst: kwadraat,
        resultaat: resultaat
    };
}
var aanroep = kwadraat(5);
document.getElementById("divResult").innerHTML =
    "Het kwadraat van 5 is: " + aanroep.uitkomst;
```

**Anonieme functies**

- Een functie die geen naam heeft is een anonieme functie
- Komt regelmatig voor
- Kan later vanuit het script niet terug aangeroepen worden
- Maak debuggen moeilijker

```
window.addEventListener("load", function() {
    alert("Alles OK!");
    // en andere nuttige zaken
});
```



Webdesign Javascript

Funcities

Arrow functies

- Zelfs het function keyword valt weg
- Herkenbaar door =>
- Zijn enkel geschikt als methode en niet als constructor (zie verder).
- Wanneer slechts 1 argument, kunnen () zelfs wegvallen.
- Wanneer slechts 1 instructie kunnen {} zelfs wegvallen.
- Als de {} wegvallen, is dit ook automatisch de return value
- this keyword (zie verder) is zelfs niet nodig.
- Werkt nog niet in Internet Explorer

```
window.addEventListener("load", () => { alert("Alles OK!"); });
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Front-end Webdesign Javascript 1-12 Funcities, Arrays en Objecten

11



Webdesign Javascript

oefening 10

Tafels van 15

- We passen oefening 9 aan.
- We maken **2 functies** aan:
 - De eerste functie **leesGetal** zal een getal tussen 1 en 20 vragen (tot het ook effectief een getal is van 1 tot 20) en teruggeven.
 - De tweede functie **toonTafels** zal de tafels van dit getal tonen.
- Voer beide functies ook meteen uit zodat je hetzelfde resultaat krijgt als bij oefening 9:

```
1 * 15 = 15
2 * 15 = 30
3 * 15 = 45
4 * 15 = 60
5 * 15 = 75
6 * 15 = 90
7 * 15 = 105
8 * 15 = 120
9 * 15 = 135
10 * 15 = 150
11 * 15 = 165
12 * 15 = 180
13 * 15 = 195
14 * 15 = 210
15 * 15 = 225
16 * 15 = 240
17 * 15 = 255
18 * 15 = 270
19 * 15 = 285
20 * 15 = 300
```

Front-end Webdesign Javascript 1-12 Funcities, Arrays en Objecten

12



Webdesign Javascript

Arrays

Array

- Array = Lijst

```
var legeArray1 = [];  
var legeArray2 = new Array();
```
- Alles kan in een array worden gestoken (var, objecten, arrays, ...)

```
var lijstGetallen1 = [10, 20, 30, 40, 50];  
var lijstGetallen2 = new Array (10, 20, 30, 40, 50);  
var diverseWaarden = ["Hello", "World", 10, 20, true];
```
- Om element uit te lezen of toe te voegen, nummer meegeven:

```
var a = lijstGetallen2[0];  
diverseWaarden[0] = "Goodbye"; // "Hello" wijzigt in "Goodbye"  
diverseWaarden[5] = false; // voegt nieuw element toe aan de array
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

13



Webdesign Javascript

Arrays

Array.length

- De eigenschap length geeft de lengte van de Array

```
var x = legeArray1.length; // x == 0  
var y = lijstGetallen1.length; // y == 5, hoogste index == 4
```
- Eigenschap kan ook toegekend worden:

```
diverseWaarden.length = 2; // diverseWaarde == ["Hello", "World"]  
diverseWaarden.length = 0; // diverseWaarde == []
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

14



Webdesign Javascript

Arrays

Array methoden

- .push() voegt een element aan het einde van array toe en geeft de nieuwe lengte terug
- .pop() verwijdert een element aan het einde van de array en geeft dit element terug
- .concat() voegt 2 arrays samen
- .slice() geeft een stuk van de array terug
- .splice() voegt elementen toe of verwijdert elementen uit de array
- .unshift() voegt elementen toe aan begin van de array, schuift andere elementen op en geeft de nieuwe lengte terug
- .shift() verwijdert eerste element uit de array en geeft dit element terug
- .join() voegt alle elementen samen tot één lange string
- .reverse() keert de volgorde van de elementen in de array om
- .sort() sorteert de elementen in de array
- ...

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

15



Webdesign Javascript

oefening 11

- Splits de functie **toonTafels** van oefening 10 in 2 afzonderlijke functies:
 - De functie **berekenTafels** plaatst de tafels 1 tot 20 van een getal in een array en geeft deze array terug.
 - De functie **toonTafels** toont nog steeds hetzelfde resultaat, maar gebruikt nu de waarde uit de array.
- Voer beide functies ook meteen uit zodat je nog steeds hetzelfde resultaat krijgt als bij oefening 9:

Tafels van 15

```

1 * 15 = 15
2 * 15 = 30
3 * 15 = 45
4 * 15 = 60
5 * 15 = 75
6 * 15 = 90
7 * 15 = 105
8 * 15 = 120
9 * 15 = 135
10 * 15 = 150
11 * 15 = 165
12 * 15 = 180
13 * 15 = 195
14 * 15 = 210
15 * 15 = 225
16 * 15 = 240
17 * 15 = 255
18 * 15 = 270
19 * 15 = 285
20 * 15 = 300
  
```

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

16



Webdesign Javascript

Arrays

Array methoden met callback function: map, filter en reduce

- Deze array methodes verwachten een functie als argument
- `.map()`: De callback functie wordt toegepast op elk element en `.map` geeft een array van al deze nieuwe elementen terug.

```
var array1 = [1, 4, 9, 16];  
var map1 = array1.map(x => x * 2); // Array [2, 8, 18, 32]  
// is identiek aan:  
// var map1 = array1.map(function(x){return x*2;});
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
<https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d>

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

17



Webdesign Javascript

Arrays

Array methoden met callback function: map, filter en reduce

- `.filter()`: De callback functie wordt toegepast voor elk element en geeft telkens een Boolean terug. Wanneer de boolean true is wordt het element overgenomen in de array die `.filter` terug geeft, in geval van false niet.

```
var words = ['spray', 'limit', 'elite', 'exuberant', 'present'];  
var result = words.filter(word => word.length > 6);  
// ["exuberant", "present"]  
// is identiek aan  
// var result = words.filter(function(word){return word.length>6;});
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
<https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d>

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

18

**Array methoden met callback function: map, filter en reduce**

- `.reduce()`: De callbackfunctie heeft 2 argumenten. Het eerste argument begint met de initiële waarde (2^e argument van `reduce`, vb1: 0) of het eerste element van de array (wanneer initiële waarde afwezig is, vb2). Vervolgens krijgt het eerste argument het resultaat van de functie van het vorige element terug. Het 2^e argument is elk element van de array. `.reduce` geeft het functie resultaat van het laatste element terug.

```
var array1 = [1, 2, 3, 4];
var total = array1.reduce(
  ( accumulator, currentValue ) => accumulator + currentValue, 0 );
// 0 + 1 + 2 + 3 + 4 = 10
var max = array1.reduce(( max, cur ) => Math.max( max, cur ) );
// 4
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce
<https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d>

**Array methoden met callback function: map, filter en reduce**

- Omdat deze functies een array teruggeven kunnen ze ook gecombineerd worden:

```
var array1 = [22, 12, 35, 4];
var totaal = array1
  .filter (getal => getal>20)
  .reduce((a,b) => a+b, 0);
// 0 + 22 + 35 = 57
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce
<https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d>



Webdesign Javascript

oefening 12

Tafels van 15

- Voeg aan oefening 11 nog een functie **toonSom1 en toonSom2** aan toe:
 - Beide functies berekenen de som van alle elementen uit de array en tonen deze.
 - toonSom1 gebruikt een for-lus om de elementen van de array op te tellen.
 - toonSom2 gebruikt de Array.reduce() functie.
 - Schrijft de callbackfunctie bij Array.reduce() zo dat dit ook werkt in Internet Explorer.

```

1 * 15 = 15
2 * 15 = 30
3 * 15 = 45
4 * 15 = 60
5 * 15 = 75
6 * 15 = 90
7 * 15 = 105
8 * 15 = 120
9 * 15 = 135
10 * 15 = 150
11 * 15 = 165
12 * 15 = 180
13 * 15 = 195
14 * 15 = 210
15 * 15 = 225
16 * 15 = 240
17 * 15 = 255
18 * 15 = 270
19 * 15 = 285
20 * 15 = 300
Som1: 3150
Som2: 3150

```



Webdesign Javascript

Objecten

Object

- Arrays en functies zijn ook speciale soorten van objecten


```
var leegObject1 = {};
```

```
var legeObject2 = new Object();
```
- Een object is container van eigenschappen
- Elke eigenschap heeft een naam en een waarde
 - De naam van een eigenschap is een String
 - De waarde kan elke waarde zijn:
 - Een enkelvoudig datatype uitz. undefined
 - Een functie
 - Een array
 - Een object

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object



Webdesign Javascript

Objecten

Object

- Het keyword `this` verwijst naar een eigenschap van het object zelf.

```
var vak = {  titel: "Front-end Webdesign Javascript",
            vakcode: "QM1153",
            fase: 1,
            lesgever: {voornaam: "Mimi", naam: "Willems"},
            volledige_naam: function {
                var volledige_naam = this.voornaam + " " + this.naam;
                return volledige_naam;
            }
        }
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

23



Webdesign Javascript

Objecten

Object

- Het keyword `this` kan ook gebruikt worden bij het aanmaken van een nieuw Object.

```
function Vak(titel, vakcode){
    this.titel = titel;
    this.vakcode = vakcode;
}
```

```
var vak1 = new Vak("Front-end Webdesign Javascript", "QM1153");
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

Front-end Webdesign Javascript 1-12 Functies, Arrays en Objecten

24



Webdesign Javascript

Objecten

Object

- Om een waarde uit te lezen of aan te passen, kunnen we zowel de . als de [] notatie gebruiken:

```
var vakTitel = vak.titel;  
var vakTitel = vak["titel"];  
vak.titel = "Front-end Webdesign Javascript";  
vak["titel"] = "Front-end Webdesign Javascript";
```

- Men kan er ook een eigenschap mee creëren:

```
vak["semester"] = 2;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer



Webdesign Javascript

Oefening 13

- Maak een object auto met als eigenschappen merk, model, aantal deuren en bouwjaar.
- Toon de eigenschappen vervolgens in de console met console.log();
- Maak hiervoor gebruik van een lus (zie 2_Webdesign Javascript Operatoren en Programma verloop) om alle eigenschappen afzonderlijk op te sommen.
- Het resultaat in de console kan er als volgt uitzien:

