Javascript deel 2

Tom Peeters

Functies

- Een functie kan je in JavaScript gebruiken om een groep opdrachten of code die regelmatig kan herhaald worden te kunnen uitvoeren met een aangegeven naar.
- De code die binnen de functie staat wordt alleen uitgevoerd wanneer de functie wordt aangeroepen.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>functie1</title>

        <script>

            function GooiDobbelsteen()
            {
                var getal = Math.round((Math.random()*5) + 1);
                alert(getal);
            }
        </script>

    </head>
    <body>
    <button onclick="GooiDobbelsteen()">Gooi met de dobbelsteen</button>
    </body>
</html>
```

Functies -2

- Het sleutelwoord function wordt gebruikt om een functie aan te maken.
- De woord wordt gevolgd door een naam van de functie (deze is zelf te kiezen).
- Na de naam codeer je een paar ronde haken ( ) => hier kan je later namen van argumenten implementeren (niet verplicht)

Functies aanroepen

- Functies doen niks op zich. De code binnen de functie (wat tussen accolades staat) wordt pas uitgevoerd wanneer de functie wordt aangeroepen
- Functies kunnen dus meerdere keren worden aangeroepen binnen hetzelfde document!

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
6      <head>
7          <title>functie1</title>
8
9          <script>
10
11             function GooiDobbelsteen()
12             {
13                 var getal = Math.round((Math.random()*5) + 1);
14                 alert(getal);
15             }
16         </script>
17
18     </head>
19     <body>
20     <button onclick="GooiDobbelsteen()">Gooi met de dobbelsteen</button>
21     </body>
22 </html>
23
```

Argumenten doorgeven

- Tot nu hebben we **1** dobbelsteen gesimuleerd. Stel we willen ook 2 dobbelstenen simuleren (dus een getal van 1 tot 12). Wat moeten we doen?

Argumenten doorgeven – een mogelijke oplossing?

- Tot nu hebben we **1** dobbelsteen gesimuleerd. Stel we willen ook 2 dobbelstenen simuleren (dus een getal van 1 tot 12). Wat moeten we doen?

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>functie1</title>

        <script>

            function GooiEenDobbelsteen()
            {
                var getal = Math.round((Math.random()*5) + 1);
                alert(getal);
            }

                function GooiTweeDobbelstenen()
            {
                var getal = Math.round((Math.random()*11) + 1);
                alert(getal);
            }
        </script>

    </head>
    <body>
    <button onclick="GooiEenDobbelsteen()">Gooi met 1 dobbelsteen</button>
    <button onclick="GooiTweeDobbelstenen()">Gooi met 2 dobbelstenen</button>
    </body>
</html>
```

Argumenten doorgeven

- Tot nu hebben we **1** dobbelsteen gesimuleerd. Stel we willen ook 2 dobbelstenen simuleren (dus een getal van 1 tot 12). Wat moeten we doen? Een betere oplossing is om een functie te schrijven die een argument kan ontvangen:

```
8
9           <script>
10
11              function GooiDobbelstenen(aantalDobbelStenen)
12              {
13                  var getal = Math.round((Math.random() *  (5*aantalDobbelStenen)+ (aantalDobbelStenen-1)) + 1);
14
15                  alert(getal);
16              }
17
18
19          </script>
20
21      </head>
22      <body>
23      <button onclick="GooiDobbelstenen(1)">Gooi met 1 dobbelsteen</button>
24      <button onclick="GooiDobbelstenen(2)">Gooi met 2 dobbelstenen</button>
25      </body>
26  </html>
```

Argumenten doorgeven

Functie **parameters of argumenten** zijn de **namen** opgelijst in de definitie van de functie.

De argumenten zijn de echte waarden ontvangen door de functie wanneer ze wordt opgeroepen

```
8
9         <script>
10
11            function GooiDobbelstenen(aantalDobbelStenen)
12            {
13                var getal = Math.round((Math.random() *  (5*aantalDobbelStenen)+ (aantalDobbelStenen-1)) + 1);
14
15                alert(getal);
16            }
17
18
19        </script>
20
21    </head>
22    <body>
23    <button onclick="GooiDobbelstenen(1)">Gooi met 1 dobbelsteen</button>
24    <button onclick="GooiDobbelstenen(2)">Gooi met 2 dobbelstenen</button>
25    </body>
26 </html>
```

Pass by reference / Pass by value

In Javascript zijn variabele complexe types (object, array) of primitieve types (integers,..). Wanneer een complex argument wordt doorgegeven als argument wordt dit “by reference” doorgegeven.(in plaats van een copy, stuurt javascript een pointer (= plaats van de locatie in het geheugen) door.

Maar wanneer een primitief type wordt doorgegeven zal javascript de value doorgeven.

```
<script>

    var objTom = {
        name:"tom",
        body:"blabla"
    };

    testRef(objTom);
    alert(objTom.name);

    function testRef(obj){
        obj.name = "peeters";
    }

    function testValue(getal){
        getal = 2;
    }

    var num = 1;
    testValue(num);
    alert(num);


</script>
```

Waarden retourneren

- Een functie kan ook een waarde retourneren (teruggeven) naar de plaats van aanroep
- Met het sleutelwoord return geef je aan welke waarde je wenst te retourneren
- (Deze geretourneerde waarde kan je dan aan een variabele toewijzen)

```
5  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
6  <head>
7      <title>return</title>
8
9  <script>
10
11     function dubbel(getal)
12     {
13         var resultaat = getal * 2;
14         return resultaat;
15     }
16 </script>
17 </head>
18
19 <body>
20     <h1> oefening op return</h1>
21     <script>
22
23         var x = dubbel(10);
24         document.write(x);
25
26     </script>
27
28
```

Waarden retourneren - 2

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title>return</title>

<script>

    function dubbel(getal)
    {
        //Kan je korter schrijven:
        return getal * 2;

    }
</script>
</head>

<body>
    <h1> oefening op return</h1>
    <script>

        //kan je rechtstreeks wegschrijven
        document.write( dubbel(10) );

    </script>


</body>
</html>

```

Waarom keyword var gebruiken? => SCOPE

```
<script>


    function f(){

        y = 9;
        var x = 8;
    }

    f();
    console.log(y);
     console.log(x);
 </script>
```

```
9                                                    keywordVAR.html:19
▶ Uncaught ReferenceError: x is not defined          keywordVAR.html:20
    at keywordVAR.html:20
>
```

Zonder declaratie is de variabele altijd global scope

```
console.log(b);
console.log(a);
a = 10;
var b = 11;
```

```
undefined   keywordVAR.html:47
▶ Uncaugh   keywordVAR.html:48
t ReferenceError: a is not
defined
    at keywordVAR.html:48
```

Waarom keyword var gebruiken?

```
<script>

    var globalvar1 = 10;
    globalvar2 = 20;


    console.log(globalvar1 + " " + globalvar2);


    function test(){
        console.log("in function test");
        console.log(globalvar1 + " " + globalvar2);

        var localvar1 = 30;
        globalvar3 = 40;

        console.log(localvar1 + " " + globalvar3);

    }

    console.log(localvar1 + " " + globalvar3);  // error: localvar werkt niet

    test();

</script>
```

Oefening

Ik wil de faculteit van een getal berekenen.

Bijvoorbeeld de faculteit van 5 = 1*2*3*4*5, de faculteit van 7 = 1*2*3*4*5*6*7

Schrijf een functie zodat je de faculteit van een willekeurig getal kan berekenen:

```
var getal =  faculteit(10);
```

Declaration function & Expression function

Markdaggett.com/blog => functions explained

2 type functie vaak gebruikt in JavaScript.

Verschil is wanneer ze geëvalueerd worden: de functie declaratie wordt aangeroepen door de interpreter wanneer deze de code parst.

De functie expressie – deel van de assignment expressie- wordt pas geëvalueerd wanneer het programma de assignment tegenkomt.

```
<script>

    MijnFunctie(5);

    function MijnFunctie(val){
        alert(val*2);
    }

    double(2);

    //anonymous function
    var double = function(val){
        alert(val*2);
    };


</script>
```

Function expression

```
myFunc();

function myFunc(){
console.log("hi func");
}
```

IN JS: functies zijn “first class objects”
⇒ Ik kan functies toewijzen aan variabelen, wat een function expressie wordt genoemd

```
var expression = function(){
        console.log("hi expression");
}
```

Wat gebeurt er nu?

```
expression();
var expression = function(){
        console.log("hi expression");
}
```

=>expression is nog geen functie, want het is nog niet toegewezen

Built-in functies (string functies)

indexOf() retourneert de positie van het eerste voorkomen van de gezochte tekst in een string

```
var str1 = "Hallo ik ben Tom Peeters";
var positie = str1.indexOf("Tom");
console.log(positie);
```

lastIndexOf() retourneert de positie van het laatste voorkomen van de gezochte tekst in een string

```
var str2 = "Hallo ik ben Bond, James Bond";
var positie = str2.lastIndexOf("Bond");
console.log(positie);
```

Ze retourneren -1 wanneer de tekst niet gevonden wordt

Built-in functies (string functies)

slice(start,end) zoekt een deel van de string op en retourneert dit.

```
var str3 = "Tom, Mieke, Hannes, Arno";
var res1 = str3.slice(12,18);
var res2 = str3.slice(20,24);
console.log(res1 + " " + res2);
```

lastIndexOf() retourneert de positie van het laatste voorkomen van de gezochte tekst in een string

```
var str2 = "Hallo ik ben Bond, James Bond";
var positie = str2.lastIndexOf("Bond");
console.log(positie);
```

Ze retourneren -1 wanneer de tekst niet gevonden wordt

Built-in functies (string functies)

charAt retourneert het karakter van de gespecifieerde positie.

```
var str = "HELLO WORLD";
str.charAt(0);
console.log(str.charAt(0));
```

length

```
var str = "woord";
console.log(str.length);
```

Oefening:

Maak een script die het aantal klinkers van een bepaald woord telt

Maak een script die de eerste letter van een woord in een hoofdletter zet (toUpperCase())

Oefeningen