Javascript basics

Xavier Dekeyster
@ksafke
http://www.linkedin.com/in/xavierd

Wat is javascript (js)?

- Ingebouwd in Netscape 2 door Brendan Eich in 1995.
- Manier om HTML en CSS aan te passen binnen een webpagina.
- Mogelijkheid om te rekenen met variabelen, getallen en datums.
- Kan ook server-side gebruikt worden (als vervanger voor bijvoorbeeld PHP). Zie node.js.

Hoe js uitvoeren?

- Aanroepen extern .js bestand: <script src="/pad/naar/bestand.js"></script>
- Tussen <script> en </script>.
- Als event-attribuut op een willekeurig attribuut. [deprecated]
- Via een javascript: protocol (in plaats van http://) in een href-attribuut van een link, of de adresbalk. [bad code]
- In de console van een webbrowser.
- Met behulp van een Userscript, zie Userscripts.org.

Hoe wordt js uitgevoerd?

- Zoals we schrijven en lezen: van boven naar beneden en van links naar rechts.
- Code kan opnieuw gebruikt worden, of in een lus herhaald worden.
- Sommige code wordt alleen uitgevoerd in bepaalde gevallen.
- Veel regels code bevatten operators om bewerkingen uit te voeren met data.
- Aan het einde van een regel met operaties hoort een ; (puntkomma).

SYNTRA WEST
www.syntrawest.be

Variabelen

```
var text = "stukje tekst, altijd tussen dubbele of enkele quotes";
var number = 42;
var multiple = 5, variables = 'separated by',
    commas = true; // mag op meerdere regels
var empty;
```

= is een operator om een waarde aan een variabele toe te kennen.

Operatoren zijn de grammatica van een programmeertaal.

Commentaar

```
var code = 'something'; // comments for one line, beginning from the first
slash
/*
 Commentaar spanning multipe rows
 Comments are written in English
*/
```

Goede code vereist geen commentaar = *bullshit*

Soorten variabelen

- Strings, tekst tussen ' of ".
- Getallen: 0.8, 42.
- Booleaanse waarden: true en false.
- Arrays: ['Appel', 'Peer', 'Banaan'].
- Objecten (set eigenschappen): {name: 'Henk', age: 24}.
- Functies (hierover later meer).

Variabelen gebruiken

```
var name = prompt('Wat is uw naam?');

alert('Hallo ' + name + '!');
```

prompt() en alert() zijn door de browser geïmplementeerde functies.

Met een +-operator kun je strings aan elkaar vastplakken (concat).

Arrays benaderen

```
var cities = ['Amsterdam', 'Berlijn', 'Parijs'];
alert(cities[0]); // geeft 'Amsterdam' in een box weer
alert(cities[2]); // geeft 'Parijs' in een box weer
alert(cities.length); // geeft 3 in een box weer
alert(cities.join(' - ')); // geeft 'Amsterdam - Berlijn - Parijs' in een box weer
```

Objecten benaderen

```
var person = { name: 'Xavier', age: 34 };
alert(person.name);
alert(person['name']); // is precies hetzelfde als de vorige
```

Variabelenamen

Mogen bestaan uit:

- Kleine- en hoofdletters;
- Getallen (behalve het eerste karakter van de naam);
- $ en _.

(In principe mogen een heleboel andere niet-whitespace UTF-8 karakters ook, maar doe dit liever niet.)

Er is ook een lijst met gereserveerde woorden, die mag je niet gebruiken als variabelenaam.

typeof

```
typeof 'henk'; // 'string'
typeof 8; // 'number'
typeof []; // 'object'
typeof (function() { return 'henk'; }); // 'function'
typeof typeof 8; // 'string'
```

Retourneert wat het type van de variabele is

Getallen

```
var number = parseInt(prompt('Kies een getal'), 10),
    tenTimes = number * 10;

alert('Tien keer ' + number + ' is ' + tenTimes);
```

parseInt(string, grondtal) is een ingebouwde functie.

Grondtal bij 'normale' getallen is 10, bij binair is het 2, hexadecimaal is 16.

Uitvoer van parseInt is altijd een decimaal getal.

Berekeningen

```
var x = 10;
x = x + 3; // x == 13
x = x - 1; // x == 12
x = x / 2; // x == 6
x = x * 4; // x == 24
x += 2; // x == 26  gelijk aan x = x + 2
x -= 13; // x == 13  gelijk aan x = x - 13
x++; // x == 14
x = x % 4; // x == 2  % is modulo, restgetal
```

parseInt(string, grondtal) is een ingebouwde functie.

Grondtal bij 'normale' getallen is 10, bij binair is het 2, hexadecimaal is 16.

Uitvoer van parseInt is altijd een decimaal getal.

++x / x++

```
var x = 5, y = 5;

document.write(++x); // 6
document.write(x); // 6
document.write(y++); // 5
document.write(y); //6
```

Bij y++ wordt de oude waarde onthouden en geretourneerd.

Bij ++x wordt de nieuwe waarde geretourneerd en niets onthouden.

Logische operatoren

var x = 6, y = 8;

- x < 10 is true want x is kleiner dan 10.
- y >= 8 is true want y is groter dan of gelijk aan 8.
- x == 3 is false want x is niet gelijk aan 3.
- x != 6 is false want x is gelijk aan 6.
- x === 6 is true want x is gelijk aan 6 *en* de types komen overeen.
- x === '6' is false want x is gelijk aan 6 *maar* de types komen niet overeen (typeof is ongelijk voor beide kanten).

Logische operatoren koppelen

Logische operatoren koppelen

var x = true, y = false;

- x && y is false want x en y zijn niet allebei true.
- x || y is true want een van beide waarden is true.
- !y is true want het omgekeerde van false is true.

Controle structuren

```
var name = prompt('Wat is uw naam?');
if (name == 'Xavier') {
  alert('Hallo cursusleider!');
}
else if (name == 'Pieter') {
  alert('Hallo organisator!');
}
else {
  alert('Hallo cursist ' + name + '!');
}
```

Brackets ({ en }) omsluiten wat uitgevoerd moet worden.

Ternary operators

Volgende code

```
var age = prompt('Je leeftijd?');
if (age < 30)
  var text = 'You probably don\'t know the Beatles.';
else
  var text = 'Did you like the beatles?';
alert(text);
```

Kan herschreven worden als

```
var age = prompt('Je leeftijd?');
var text = (age < 30) ? 'You probably don\'t know the Beatles.' : 'Did you like
the beatles?';
alert(text);
```

True or false

```
if (10 == 10) { // true
if (9 == 10) { // false
if (null) { // false
if (NaN) { // false
```

De for lus

```
for (var i = 1; i < 7; ++i) {
  document.write('<h' + i + '>Tekst</h' + i + '>');
}
```

- var i = 1 wordt uitgevoerd *voor* men in de lus gaat.
- De logische expressie i < 7 wordt uitgevoerd elke keer voor iets herhaald wordt: is het false, dan wordt de lus onderbroken.
- ++i wordt uitgevoerd na het codeblok.

SYNTRA WEST
www.syntrawest.be

De while lus

```
var name = prompt('Wat is uw naam?');
while (name != 'Xavier') {
  alert('Naam is ongeldig, kies een andere.');
  name = prompt('Wat is uw naam?');
}
alert('Eindelijk Xavier!');
```

- Lus wordt herhaald zolang name != 'Arjan' als resultaat true teruggeeft.

break en continue

Met **break** stop je de lus en gaat de JS engine verder met de rest van de code.

Met **continue** skip je tot het einde van de lus-body. De lus wordt vanaf daar hervat.

```
for (var i = 1; i < 100; ++i) {
  if (i > 5 && i < 10)
    continue;
  if (i > 15)
    break;
  document.write('<p>Teller is nu ' + i);
}
document.write('<p>Waarde van i ' + i);
```

Functies

Functies zijn gegroepeerde acties die meerdere malen aangeroepen kunnen worden.

```
function myName(name) {
  document.write('Mijn naam is: ' + name + '<br>');
}
myName('Xavier');
myName('Pieter');
```

De functie vraagt één parameter: name, die meegegeven wordt bij het aanroepen van de functie.

Return

Twee mogelijke soorten gebruik:

- Voortijds de functie stoppen (lijkt op break);
- Een waarde retourneren.

```
function sum(number1, number2) {
  return number1 + number2;
}

var something = sum(4, 5); // something contains 9 as value
```

Arguments

arguments is een array met alle argumenten.

```
function sum() {
  var toReturn = 0;
  for (var i = 0; i < arguments.length; ++i) {
    toReturn += arguments[i];
  }
  return toReturn;
}

var something = sum(4, 5), // something contains 9 as value
    somethingElse = sum(1), // somethingElse contains 1 as value
    more = sum(1, 2, 3, 4, 5); // more contains 15 as value
```

Variabele scope

Variabelen en functies zitten in de globale scope. Echter, als je een functie maakt is er ook een lokale scope. Variabelen daarbinnen aangemaakt zijn niet van buitenaf aan te passen of aan te roepen.

```
var name1 = 'Xavier';
function sayName() {
  alert('Voor: ' + name1);
  name1 = 'Piet';
  alert('Tijdens: ' + name1);
}
sayName();
alert('Na: ' + name1);
```

```
var name2 = 'Xavier';
function sayName() {
  alert('Voor: ' + name2);
  var name2 = 'Piet';
  alert('Tijdens: ' + name2);
}
sayName();
alert('Na: ' + name2);
```

SYNTRA WEST

www.syntrawest.be

Anonieme functies

Functies hoeven niet perse een naam te hebben.

```
var elm = document.getElementById('main');
elm.onclick = function() {
  alert('Hallo!');
}
```

Zelfuitvoerende anonieme functies

Door een anonieme functie aan te roepen door er een () achter te plakken, wordt gelijk de inhoud van de anonieme functie uitgevoerd.

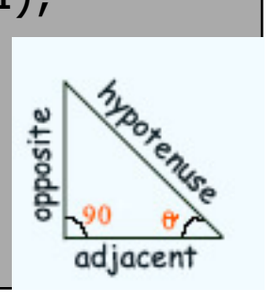
Dit is handig omdat de variabele scope zich in de functie bevindt. Er wordt helemaal niet meer naar de globale scope geschreven of gelezen.

```
(function(){
  alert('Hallo!');
})();
```

Math

Met het Math object kun je krachtige mathematische functies aanroepen.

```
function getAngle(opposite, adjacent) {
  return Math.round((Math.atan(opposite / adjacent) * 180) / Math.PI);
}
alert(getAngle(1, 1));
```



Date

Met Date kun je met datums rekenen.

```
var start = (new Date()).getTime();
// do stuff
var end = (new Date()).getTime();
alert((end - start) / 1000);
```

new Data maakt een nieuw object aan. Dit object heeft diverse functies die je aan kunt roepen. getTime() is een functie die het aantal milliseconden sinds 1 januari 1970 teruggeeft.

Times en intervallen

Het is mogelijk om een functie na een bepaalde tijd, of met een bepaald interval, uit te voeren. Dit is essentieel voor het maken van animaties. (Er is voor animaties nu iets beters, requestAnimationFrame, maar dat terzijde.) Het is mogelijk de intervallen te clearen als toch blijkt dat je die niet meer nodig hebt.

```
function say(text) {
  document.write('<p>' + ((text && text.length > 0) ? text : 'Hello!') + '</p>');
}
setInterval(say, 1000);
var myTimer = setInterval(function(){say('A good day to you!')}, 2000);
setTimeout(function() {
  clearInterval(myTimer);
}, 10000);
```

Voorbeeld

Opdracht: vraag de gebruiker om een tekst. Plaats de letters op het scherm met verschillende hoogtes en kleuren, zodat het er uitziet alsof een kind het heeft gemaakt.

```
var text = prompt('Some text please?'),
    str = '',
    colors = ['red', 'green', 'blue', 'purple', 'yellow'];
for (var i = 0; i < text.length; ++i) {
  var color = colors[Math.floor(Math.random() *
colors.length)],
      size = 22 + Math.round(Math.random() * 20);
  str += '<span style="color:' + color + ';font-size:' + size +
'px;font-weight:bold">' + text.charAt(i) + '</span>';
}
document.write(str);
```

Oefeningen

Kwadraat

Loop alle getallen van 1 tot 400 en print het getal op het scherm indien de waarde een kwadraat is.

Hint

Er bestaat een functie Math.sqrt(number) die de wortel van number retourneert.

Oefeningen

Som

Vraag aan de gebruiker een getal groter dan nul. Tel alle getallen van 1 tot dat getal op en geef deze som terug in een alertbox.

Oefeningen

Woonkamer

Schrijf een functie waarin je lengte, breedte en hoogte van een kamer meegeeft en die de oppervlak van de muren teruggeeft. Maak ook een werkend proramam zodat je met drie prompts een alert krijgt met de totale oppervlakte. (Handig als je wilt schilderen.)

Oefeningen

Vergelijk abonnementen

Schrijf een script dat vraagt hoeveel telefoonabbonnementen je wilt vergelijken. Maak een array aan met die lengte, en vraag door middel van prompt wat de eenmalige prijs, de maandelijkse prijs en het aantal maanden zijn. Geef daarna de goedkoopste terug.

Hint

Sorteren van een array kan gemakkelijk met .sort. Een multidimensionale array sorteer je als volgt:

```
var myArray = [['Xavier', 34], ['Piet', 23], ['Henk', 1]];
myArray = myArray.sort(function(a, b) {
  return a[1] - b[1];
});
alert(myArray[0][0]);
```

a en b zijn de arrays binnen myArray. Als de functie een negatief getal teruggeeft, is a lager dan b. Bij 0 zijn ze gelijk. Bij een positief getal is a hoger dan b.