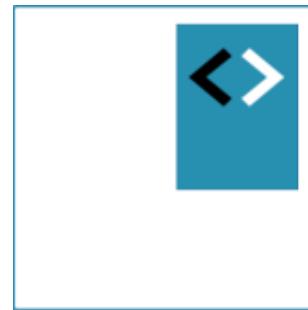




Ministerie van Defensie

HTML, CSS en JavaScript

Module : JavaScript



Peter Kassenaar – info@kassenaar.com

Peter Kassenaar – info@kassenaar.com



Kennismaken met JavaScript

Gedrag, logica en 'intelligentie' toevoegen aan
webpagina's.



Java heeft net zo veel met
JavaScript te maken,
als Ham met Hamburger



1. History & Future

Where do we come from, where are we going?

1995





NETSCAPE®



Brendan Eich

10

days

Mocha

LiveScript

JavaScript



ECMAScript

JavaScript / ECMAScript	Release date
1.0	1996
1.2	1997
1.5	1999/2000
ES3	1999
ES5	2009
ES6 / ECMAScript 2015	2015
ECMAScript Next (7)	2016
...	June, every Year

JavaScript status



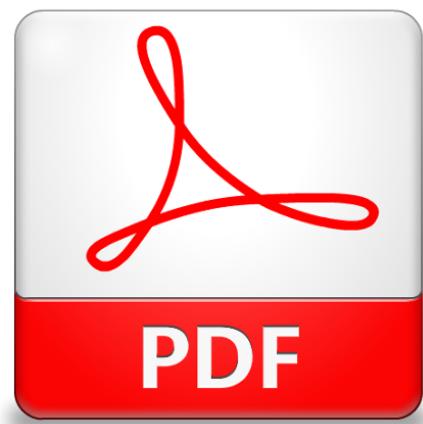
2006



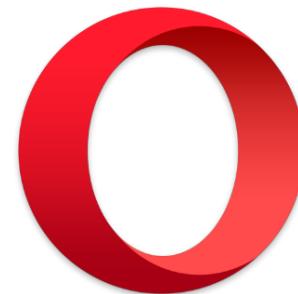
2016-



Open Standard



Mostly...



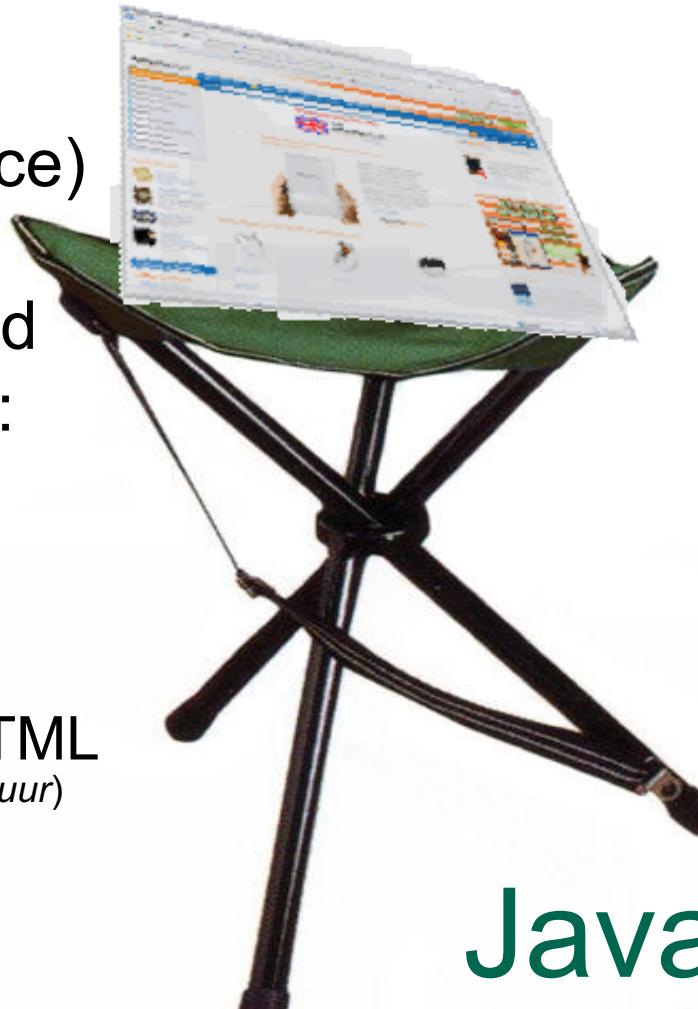
Uitgangspunten

- Javascript gebruiken in webapplicaties
 - de browser is de container waarbinnen het script wordt uitgevoerd
- Doel: functionaliteit en gebruiksvriendelijkheid
- In combinatie met andere technieken



De plaats van JavaScript

De site
(User Interface)
wordt
samengesteld
op basis van:



(X)HTML
(structuur)

CSS
(presentatie)

JavaScript/DOM
(gedrag)

Uitgangspunten

- JavaScript zelf (EcmaScript) is de **taal** (*core language*)
- JavaScript kent verschillende toepassingsgebieden
 - Client sided (browser)
 - Server sided (webserver)



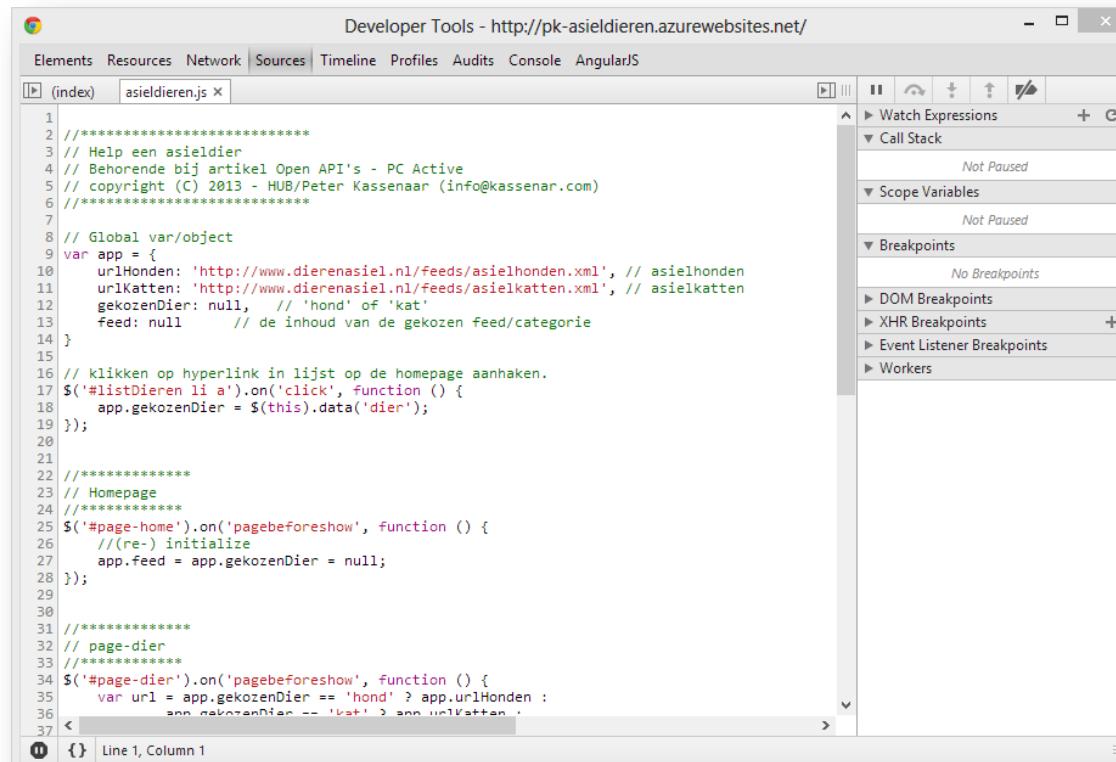
Ook server sided JavaScript

- node.js
- doel: schaalbare serverapplicaties bouwen
- Valt buiten bestek van deze cursus



Debugging – ingebouwd in de browser

- Chrome – Developer Tools
 - Firefox – Firebug
 - Internet Explorer – F12 Tools



Snel en veel voorbeelden: W3Schools

The screenshot shows the W3Schools website with the URL <https://www.w3schools.com/js/default.asp>. The page title is "JavaScript Tutorial". The left sidebar contains a navigation menu for "JS Tutorial" with many items listed, including "JS HOME" which is highlighted. The main content area features a green box with the following text:

JavaScript is the programming language of HTML and the Web.
JavaScript is easy to learn.
This tutorial will teach you JavaScript from basic to advanced.

Below this, there is a section titled "Examples in Each Chapter" with a sub-section titled "Example" containing the text "My First JavaScript".

<https://www.w3schools.com/js/default.asp>

Aanbevolen documentatie - MDN

Welcome to the new, Improved MDN design. We hope you like it! If you have any suggestions or comments, [please share!](#)

[Aanmelden met Persona](#) [mozilla](#)

 ZONES WEBPLATFORM DEVELOPER PROGRAM HULPMIDDELLEN DEMO'S [Search icon](#)

MDN > Web technology for developers > JavaScript [TALEN](#) [PAGINA BEWERKEN](#) [Settings icon](#)

JavaScript

JavaScript® (often shortened to **JS**) is a lightweight, interpreted, object-oriented language with [first-class functions](#), most known as the scripting language for Web pages, but [used in many non-browser environments](#) as well such as [node.js](#) or [Apache CouchDB](#). It is a [prototype-based, multi-paradigm](#) scripting language that is dynamic, is [type safe](#), and supports object-oriented, imperative, and functional programming styles.

The JavaScript standard is [ECMAScript](#). As of 2012, all modern browsers fully support ECMAScript 5.1. Older browsers support at least ECMAScript 3. A 6th major revision of the standard is in the works. The current progress of different new and improved features can be followed on the [dedicated wiki](#).

This section of the site is dedicated to the JavaScript language itself, the parts that are not specific to Web pages, or other host environments. For information about APIs specific to Web pages, please see [DOM](#). Read more about how DOM and JavaScript fit together in the [DOM Reference](#).

Documentation

[JavaScript Reference](#)
This JavaScript reference includes complete documentation for JavaScript 1.5 and updates.

[JavaScript Guide](#)
Our primary guide about how to program with JavaScript, for programmers who are new to JavaScript.

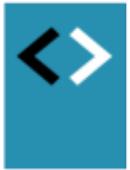
Introductory articles

[JavaScript technologies overview](#)

Tools & advanced resources

- [LearnStreet](#) - Free online JavaScript tutorials and practice exercises.
- [Codecademy](#) - Free JavaScript course with interactive problems
- [Code School](#) - Learn by Doing, Multiple JS courses
- [Idiomatic.js](#) - Principles of Writing Consistent, Idiomatic JavaScript
- [Memory Management in JavaScript](#) . Overview of how memory management works in JavaScript
- [Firebug](#) - JavaScript debugging and profiling
- [Venkman](#) - JavaScript debugger
- [JavaScript Shells](#) - test snippets of code

<https://developer.mozilla.org/nl/docs/Web/JavaScript>



2. JavaScript Core

Beginnen met zelf JavaScript schrijven

Peter Kassenaar

JavaScript

Web Development Library



VAN DUUREN INFORMATICA

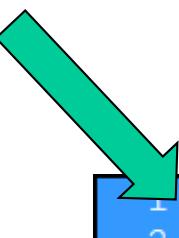
Hoofdstuk 2 – Statements,
gegevenstypen en variabelen
p.30 e.v.

1. JavaScript in de pagina

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Mijn eerste JavaScript</title>
6 <script>
7   alert ('Hello world');
8 </script>
9 </head>
10
11 <body>
12 </body>
13 </html>
14 |
```

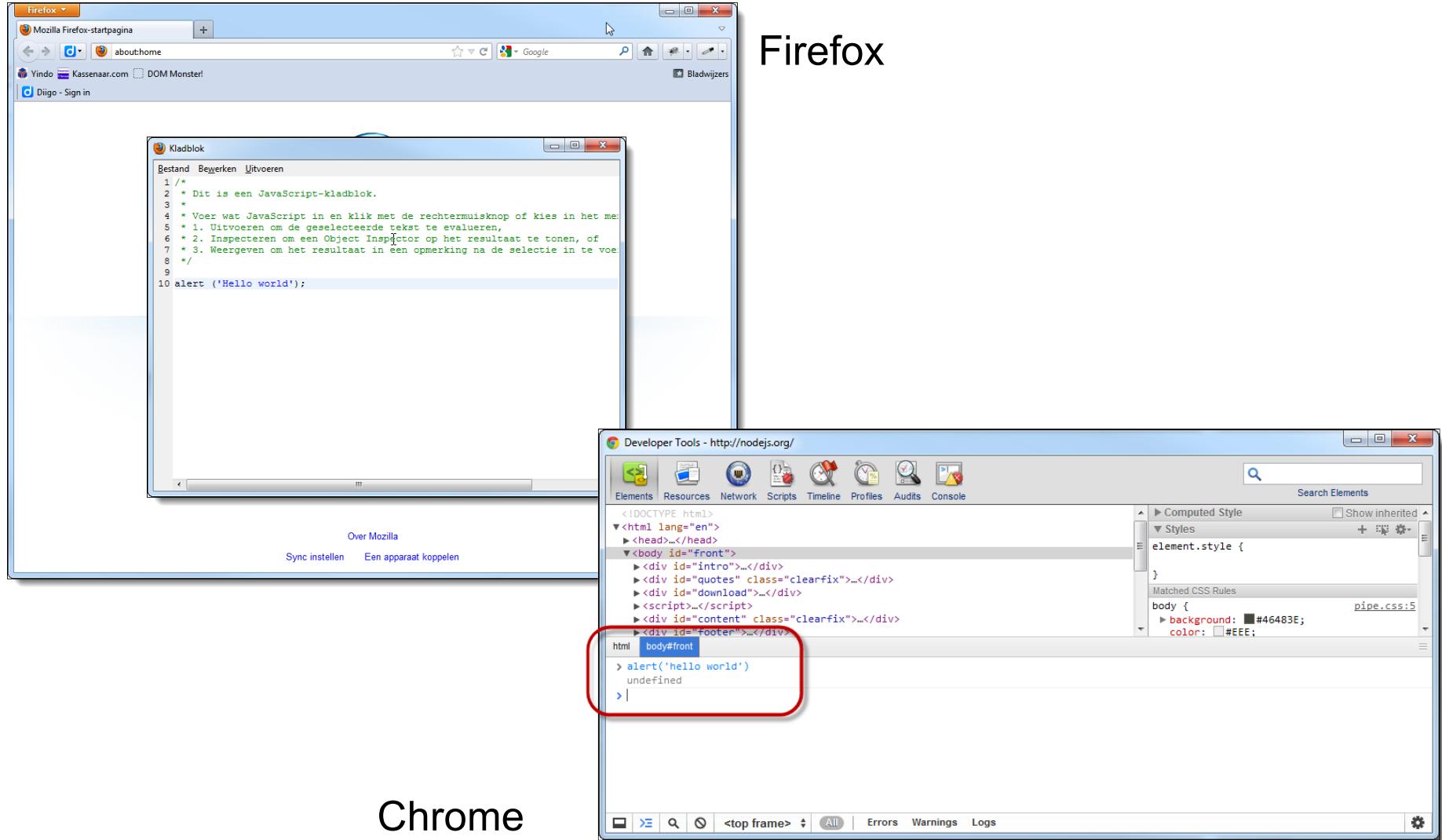
2. Gekoppeld scriptbestand

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Mijn eerste JavaScript</title>
6 <script src="scripts/02_helloWorld.js">
7 </script>
8 </head>
9
10 <body>
11 </body>
12 </html>
13 |
```



```
1 // JavaScript Document
2 alert('Hello world');
```

3. Console



Syntaxis

- Let op het gebruik van hoofd- en kleine letters
 - JavaScript is *case sensitive*
- Gebruik het keyword `var` om variabelen te declareren
- Variabelen hebben geen vast 'type'
 - impliciete casting

```
1 // JavaScript Document
2 var getal = 'Hallo';
3 getal = getal + 1;
4 alert(getal);
```

Syntaxis

- De waarden van strings staan tussen ‘...’ of “...”
- Beëindig elk statement met puntkomma ;
- Commentaar met
 - // één regel
 - /* ... */ meerdere regels

```
1 // JavaScript Document
2 //var getal = 'Hallo';
3 //getal = getal + 1;
4 //alert(getal);
```

Syntaxis

- Haakjes : voorrangsregels

- `(5 + 2) * 3 ; // uitvoer: 21`

- Accolades : statements groeperen

```
if (...) {  
    // overige programmacode  
}
```

- Meer accolades: objecten definiëren

```
{  
    Property1 : value1,  
    propertyN : valueN  
}
```

- Inspringen / witruimte

- niet verplicht – maakt wel de code leesbaarder.
 - Maak een keuze voor spaties of tabs

Operatoren

- Een operator *doet* iets.
- Belangrijk: *Operator Precedence*
 - welke operator wordt eerst uitgevoerd
 - 'Meneer Van Dalen Wacht Op Antwoord
 - ...maar dan anders
- Operator Families

with the expected result that `a` and `b` get the value 5. This is because the assignment operator returns the value that it assigned. First, `b` is set to 5. Then the `a` is set to the value of `b`.

Table

The following table is ordered from highest (1) to lowest (18) precedence.

Precedence	Operator type	Associativity	Individual operators
1	new	right-to-left	new
2	function call	left-to-right	()
3	increment	n/a	++
	decrement	n/a	--
4	logical-not	right-to-left	!
	bitwise not	right-to-left	~
	unary +	right-to-left	+
	unary negation	right-to-left	-
	typeof	right-to-left	typeof
	void	right-to-left	void
	delete	right-to-left	delete
5	multiplication	left-to-right	*
	division	left-to-right	/
	modulus	left-to-right	%
6	addition	left-to-right	+
	subtraction	left-to-right	-
7	bitwise shift	left-to-right	<< >> >>>
8	relational	left-to-right	< <= > >=

<https://www.google.nl/url?sa=t&rct=j&q=&esrc=s&source=web&cd...>

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

Families operatoren

- Arithmetic: +, -, *, /, %
 - wiskundig (optellen, aftrekken, etc)
- Assignment: =, +=, -=, *=, /=, %=
 - toekennen
- Comparison: ==, !=, >, >=, <=, ===, !==
 - Vergelijkingen

- Boolean: !, && , ||
 - Vergelijkingen
- Conditionals:
 - *variable = boolean_expression*

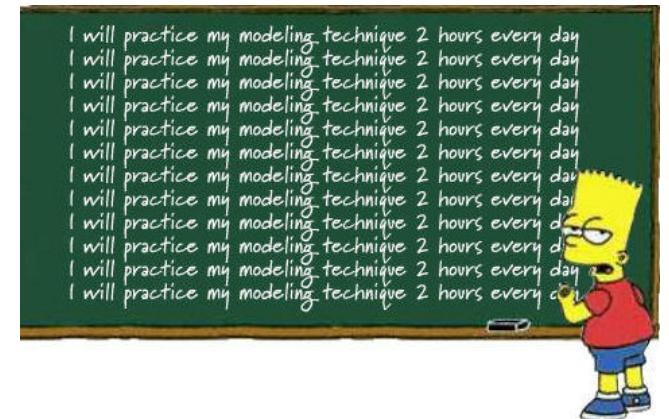
? *true_value*

! *false_value;*

```
6 <script>
7 var a = 10;
8
9 var b= (a > 10)
10    ? 'a is groter dan tien'
11    : 'a is tien of minder';
12
13 alert (b);
14
15 </script>| I
16 |>
```

Oefening

- schrijf een script dat twee variabelen bij elkaar optelt en het resultaat in een alert-venster toont.
 - Alternatief: gebruik `console.log('...')`
- Voorzie je script van nuttig commentaar.
- Schrijf een statement *met* en zonder haakjes.
 - bekijk de verschillen in uitvoer
 - bv `3 + 4 * 8` vs `(3 + 4) * 8`



Statements

- Doel: programmaflow beïnvloeden
- Typen:
 - Conditionele statements
 - Loops
 - compound statements

Conditionele statements

- Doel – testen op een bepaalde voorwaarde (*conditie*)
 - if ... else
 - switch

If ... else

- andere naam: *voorwaardelijke statements*
- Vaak : gebruik van comparison operators
 - <, <=, ==, >, >=
- Vaak : gebruik van boolean operators
 - !, &&, ||

```
6 <script>
7 var a = 10;
8 var b= '';
9
10 if(a > 10){
11     b = 'a is groter dan tien' ;
12 }
13 else{
14     b= 'a is gelijk aan tien of kleiner dan tien';
15 }
16
17 alert (b);
18
19 </script>
```

If ... else

- Else gebruik je om alternatieven weer te geven
- Kan gecombineerd worden
 - else if()....

```
7 var a = 10;
8 var b= '';
9
10 if(a > 10){
11     b = 'a is groter dan tien' ;
12 }
13 else if (a==10){
14     b= 'a is gelijk aan tien';
15 }
16 else{
17     b= 'a is kleiner dan tien';
18 }
19
20 alert (b);
```

Lussen

- while
- do ... while
- for
- for ... in
- with
- label, break and continue
- try ... catch ... finally
- throw
- Je hoeft eigenlijk alleen for() te kennen!
- De rest is 'nice to have'

For-lus

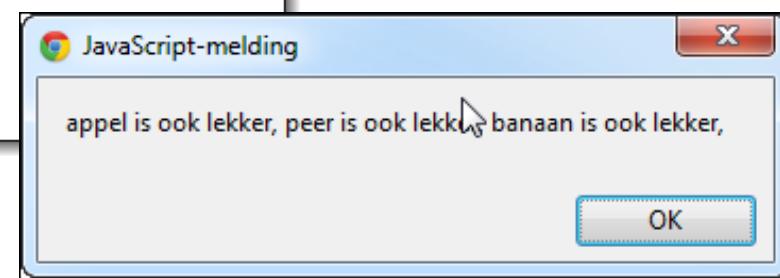
- Loop over een aantal variabelen zolang, of totdat aan een bepaalde voorwaarde is voldaan

```
for (var i = 1; i <= 10; i++) {  
    //... voer berekening uit...  
}
```

For / for in

- Eveneens voor herhaling acties
- Vaker gebruikt dan while
- Vaak toegepast op arrays of verzamelingen:

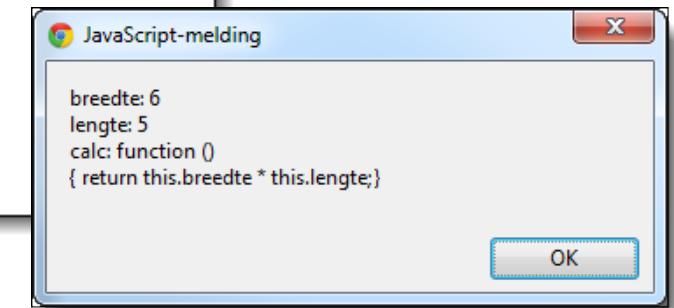
```
6 <script>
7 // variabelen
8 var bericht = '';
9 var i;
10 var mijnArray = ['appel', 'peer', 'banaan'];
11
12 for (i = 0 ; i < mijnArray.length; i++){
13     bericht += mijnArray[i] + ' is ook lekker, |';
14 }
15
16 alert (bericht);
17
18 </script>
```



for...in

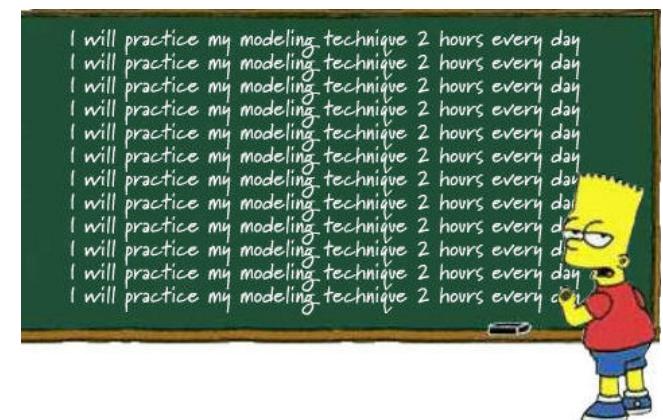
- For...in wordt gebruikt om members van een object te bereiken

```
6 <script>
7 // variabelen
8 var bericht = '';
9 var i;
10 var mijnObject = {
11     breedte: 6,
12     lengte : 5,
13     calc : function()
14         { return this.breedte * this.lengte; }
15 }
16 for (i in mijnObject){
17     bericht += (i + ': ' + mijnObject[i]+ '\n');
18 }
19
20 alert (bericht);
21
22 </script>
```



Oefening

- Schrijf een script dat de tafel van 3 in een alert (of in de console) toont..
 - Gebruik hiervoor een for-lus.





Complexe variabelen

Functies, Arrays en Objecten

Syntaxis – complexe variabelen

- Drie ‘complexe’ typen variabelen

- function
- array
- object

```
6 <script>
7 // Complexe typen variabelen
8
9 // 1. Array:
10 var mijnArray = ['appel', 'peer', 'banaan'];
11
12 // 2. object
13 var mijnObject = new Object();
14 mijnObject.voornaam = 'peter';
15
16 // 3. function
17 function optellen(getal1, getal2){
18     var resultaat = getal1 + getal2;
19     return resultaat;
20 }
21
22 </script>
```

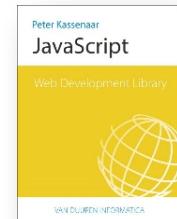


Functies

Stukken code meerdere malen, op verzoek, kunnen uitvoeren

Complex gegevenstype 1 - Functies

- Belangrijk programmeerprincipe: DRY
- Functies helpen om vaak voorkomende taken maar een keer te moeten programmeren
- Een functie...
 - doet uit zichzelf niks (wordt *aangeroepen* of *invoked*)
 - collectie van statements die onbeperkt kunnen worden aangeroepen



p.72 e.v.

Het keyword function

- Definiëren: gebruik het keyword `function`
- naamgeving – wees duidelijk en beknopt
- Let op de syntaxis: haakjes en accolades

```
6 <script>
7 //1. functiedefinitie
8 function kwadraat(x) {
9     return x * x;
10 }
11
12
13 // 2. functieaanroep| ←
14 alert ('Het kwadraat van 19 is ' + kwadraat (19));
15
16 </script>
```

Eigenschappen van functies

- Functies kunnen een waarde retourneren
- Functies kunnen tot 255 argumenten of 'parameters' accepteren
- De naam van parameters mag je zelf verzinnen (hier: a en b)

```
// 1. Een JavaScript
calculator:
function optellen(a, b) {
    return a + b;
}
```

```
// 2. aanroepen:
var uitkomst = optellen(10, 20); // 30
```

Waarden retourneren

- In principe wordt 1 waarde teruggegeven
- Meerdere waarden retourneren?
 - Wrappen in een object

```
function myManyReturnFunction(number1, number2) {  
    return {  
        x: number1 * number2,  
        y: number1 / number2  
    }  
}
```

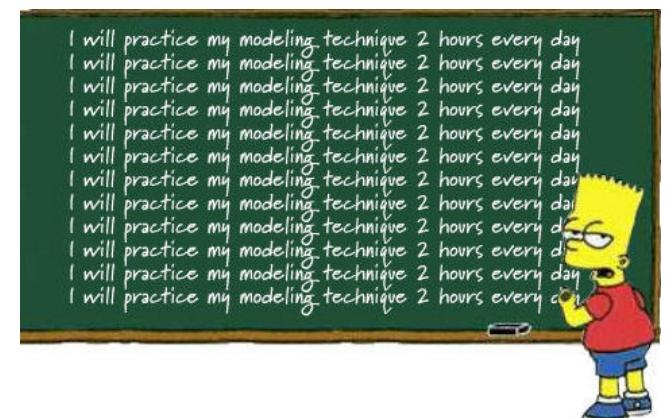
Anonieme functies

- Meer uitgebreide scripts: vaak *anonieme functies*
 - hebben geen naam
 - Veel toegepast: callback-statements
 - Doel: automatisch uitvoeren

```
6 <script>
7 //1. Anonieme functie - wordt automatisch uitgevoerd
8 window.onload=function() {
9     var mijnDiv = document.getElementById('div1');
10    mijnDiv.style.backgroundColor ='#000';
11 }
12 </script>
13 <style>
14 /* stijlen */
15 #div1{
16     width:100px;
17     height:100px;
18     border: 2px solid red;
19 }
20 </style>
21 </head>
22 <body>
23 <div id="div1"></div>
24 </body>
```

Oefening

- Schrijf vier functies die respectievelijk twee getallen optellen, aftrekken, vermenigvuldigen en delen
- Schrijf in de deelfunctie een controle op het getal 0 in de noemer



Complex gegevenstype 2 - Arrays

- Een array is een *verzameling* van variabelen
 - primitieven
 - complexen
 - of een combinatie
- Een array kan dus andere arrays bevatten
- Het benoemen van elementen in een array begint bij 0.

Codevoorbeeld array

```
6 <script>
7 // Verschillende notatiewijzen voor arrays
8
9 // 1. Array:
10 var mijnArray = ['appel', 'peer', 'banaan'];
11
12 // 2. Andere notatiewijze, zelfde resultaat
13 var mijnFruit = new Array('appel', 'peer', 'banaan');
14
15 // 3. Derde notatiewijze, zelfde resultaat
16 var nogMeerFruit = new Array(3);
17 nogMeerFruit [0] = 'appel';
18 nogMeerFruit [1] = 'peer';
19 nogMeerFruit [2] = 'banaan';
20
21 alert (nogMeerFruit[2]); // uitvoer: banaan
22
23 </script>
```

Methods voor array

- `.join('...')` voegt array samen naar string, met optionele separator
- `.split()` splitst een string in een array, op basis van optioneel scheidingsteken.
- `.toString()`
 - als `.join()`, maar zonder separator-keuze
- `.pop()` verwijdert element aan einde array
- `.push()` voegt element toe aan einde van array
- `.shift()` - haalt element weg van begin van array



p.72 e.v.

Array.forEach()

Arrayfunctie forEach () {...}

Alternatief voor

```
for (var x = 0; ... ; ...) { ... }
```

1. - Neemt een functie als parameter
2. - functie wordt uitgevoerd voor elk item in de array.
3. - Alleen IE9 en hoger (=volledig ES5-compliant browsers)

Complexe: meer array-transformaties

- `Array.filter()`
- 1. - Neemt eveneens een functie als parameter
 2. - functie wordt uitgevoerd voor elk item in de array.
 3. - Retourneer de waarde die voldoet aan je voorwaarde, wordt opgeslagen in nieuwe array
- `Array.map()`
- 1. Krijgt ook een functie als parameters;
 2. voert TRANSFORMATIE uit voor elk item in de array
- Vergeet het keyword `return` niet!

Arrayfuncties bij MDN

The screenshot shows the Mozilla Developer Network (MDN) website. The URL is https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array. The page title is "Array". The main content area starts with a brief introduction: "The JavaScript `Array` global object is a constructor for arrays, which are high-level, list-like objects." Below this is a "Syntax" section containing code examples:

```
[element0, element1, ..., elementN]
new Array(element0, element1[, ...[, elementN]])
new Array(arrayLength)
```

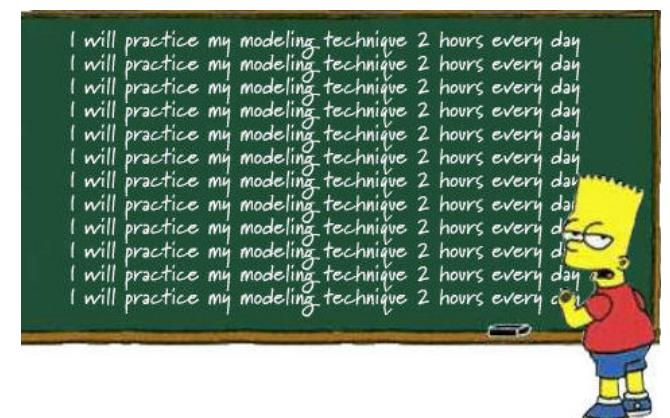
Following the syntax are two detailed sections: "elementN" and "arrayLength". The "elementN" section explains that a JavaScript array is initialized with the given elements, except in the case where a single argument is passed to the `Array` constructor and that argument is a number. The "arrayLength" section states that if the only argument passed to the `Array` constructor is an integer between 0 and $2^{32}-1$ (inclusive), it returns a new JavaScript array with length set to that number. If the argument is any other number, a `RangeError` exception is thrown.

The right sidebar, titled "IN THIS ARTICLE", lists several sections: Syntax, Description, Accessing array elements, Relationship between `length` and numerical properties, Creating an array using the result of a match, Properties, Methods, Array instances, Properties, Methods, Mutator methods, Accessor methods, Iteration methods, Array generic methods, Examples, and Creating an array.

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

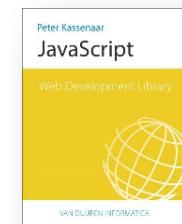
Oefening

- Schrijf een script dat een array maakt
 - toon eigenschappen v/d array
 - lengte, item op positie xx
 - Voeg via script element toe aan de array. Toon dit in alert().
- Maak twee array's
 - voeg deze samen via script
 - toon uitvoer in alert().
 - Splits een array naar een string
 - Gebruik diverse delimiters



Complex gegevenstype 3 – Objecten

- Objecten zijn als arrays, maar dan met *benoemde waarden*
- Een object is een ‘supervariabele’
 - Bij elkaar horende eenheden informatie
- kan bevatten:
 - eigenschappen
 - functies
 - andere objecten
 - arrays



Syntaxis objecten

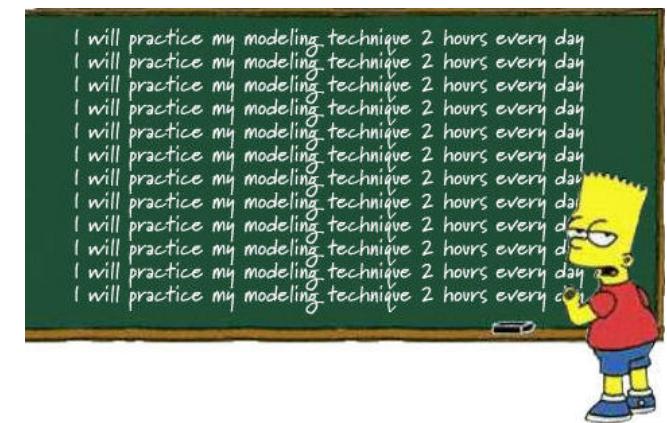
```
6 <script>
7 // Een object maken met ingekapselde method/function
8
9 // 1. Traditionele notatie
10 var o = new Object();
11 o.breedte = 5;
12 o.lengte = 4;
13 o.calc = function(){
14     var oppervlakte = this.breedte * this.lengte;
15     return oppervlakte;
16 }
17 alert (o.calc());
18
19 // 2. Moderne notatie (JSON)
20 var p = {
21     breedte: 6,
22     lengte : 5,
23     calc : function()
24         { return this.breedte * this.lengte; }
25 }
26 alert (p.calc());
27
28 // ...
```

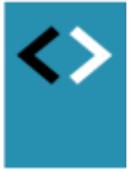
JSON = JavaScript Object Notation

Oefening

- Maak zelf een object en geef dit leden (*members*)
 - bijvoorbeeld: vervoermiddelen, je gezin, dieren...
 - gevorderd: probeer ook een functie te maken binnen het object die iets nuttigs doet
 - Roep de functie binnen het object aan
 - Test met wel/niet doorgeven parameters
 - toon de uitvoer in alert() of via
console.log()

Keep it Simple
(for now)





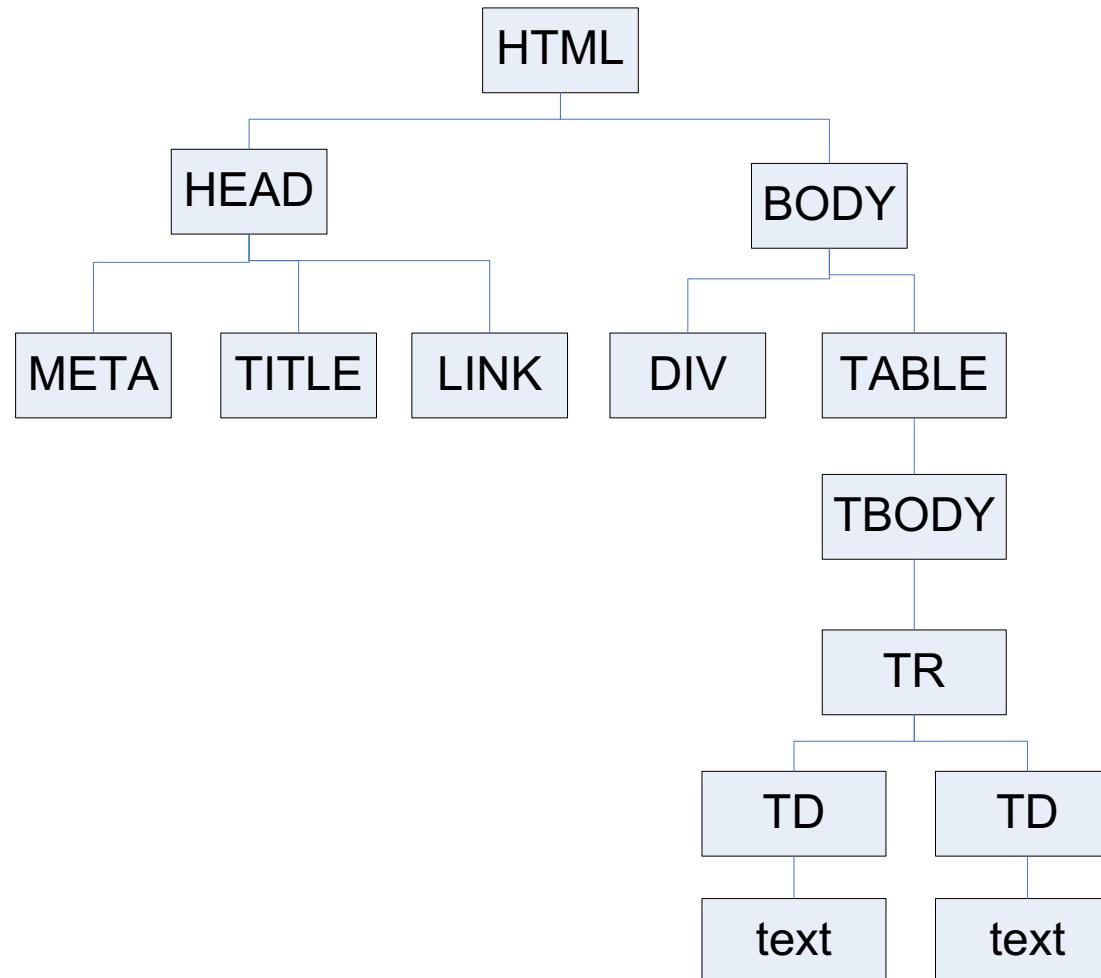
JavaScript en het DOM

Elementen in webpagina's selecteren en manipuleren

Client Sided JavaScript

- JavaScript *toegepast* in een omgeving
 - web browser
 - (maar ook: Node.js, Flash Player, Adobe Reader, e.a.)
- Maak gebruik van je Core JavaScript-kennis om concrete projecten te realiseren

Document Object Model



Document Object Model

- Ingebouwde Objectstructuur in de Browser
 - HTML elementen zijn benaderbaar via javascript
 - Dynamisch aanpassen van HTML
 - Vroeger: '*DHTML*'
 - Gestandaardiseerd door W3C (DOM level 2)
 - Verschillen per browser

Methoden voor document

- De belangrijkste methoden op het document:
 - `document.getElementById();`
 - `document.createElement();`
 - `document.createTextNode();`
 - `document.getElementsByTagName();`
 - `document.getElementsByClassName();`

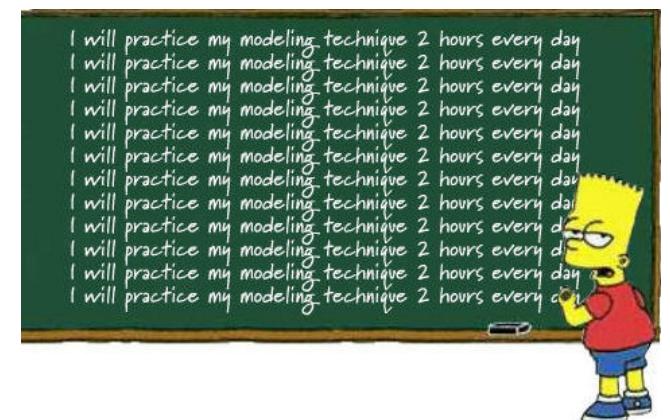
- IE9+ en moderne browsers:
 - `.getElementsByClassName()`
 - `.getElementsByTagName()`
- Belangrijk:
 - `.querySelector()`
 - `.querySelectorAll()`
 - Niet ondersteund door oudere browsers
 - Gebruik CSS-syntaxis om elementen te selecteren!
- Verouderd:
 - `document.write();`

DOM - gebruik

- Pagina manipuleren
 - informatie weghalen en toevoegen
 - eigenschappen van elementen veranderen
(zichtbaarheid, positie, kleur)
- *Webapplicaties* maken in plaats van websites
- Externe aanpassingen:
 - via AJAX communicatie met webserver
 - via events interactie met de gebruiker

Oefening

- Schrijf een script dat een alinea toevoegt aan een pagina waarop de bezoeker wordt begroet.
- begroeting is afhankelijk van tijdstip: goedemorgen, goedemiddag, goedenavond, goedenacht.
- gebruik :
 - new Date() en getHours()
 - document.createElement()
 - document.createTextNode()
 - document.getElementById('xxx')
 .appendChild();





Event Driven JavaScript

Je website of app laten reageren op gebeurtenissen

Event-driven JavaScript

- Events zijn *gebeurtenissen* in de browser
 - hetzij afhankelijk van het systeem
 - hetzij door de gebruiker geïnitieerd
- Events worden afgevuurd door (elementen op de) pagina

- Events *kunnen* worden afgevangen en aanleiding zijn tot de uitvoering van een script
- Sommige events kunnen worden afgevuurd door alle elementen (mouseover); sommige door slechts enkele elementen (change)

Events voorbeelden

- Window
 - load
 - resize
 - unload
 - scroll
- input, select, textarea
 - change
 - focus
 - blur
- button
 - focus, blur
- zichtbare elementen
 - mouseover
 - mouseup
 - mouseout
 - mousedown
 - mousemove
- alle
 - click
- ...tal van andere

Nieuw: mobile events

Photoshop PSD file d

- touch
- tap
- swipe
- pinch
- zoom
- rotate
- orientationchange



Events zijn geen HTML-attributen

- Als HTML-attribuut: (*deprecated!*)
 - onload
 - onmouseover
 - etc
 - <button onclick="doFunction()">
- Events:
 - load
 - mouseover
 - etc.
- Event handler – de event verder verwerken
 - onload
 - onmouseover

Aanbevolen: unobtrusive JavaScript

- Stel je hebt een functie:

```
function presskey(e) {  
    var key = e.keyCode;  
    alert("Je hebt gedrukt op " + String.fromCharCode(key));  
}
```

- Dan niet:

```
<textarea rows="10" cols="40" id="myText" onkeypress="presskey()></textarea>
```

- Maar:

```
<textarea rows="10" cols="40" id="myText"></textarea>
```

- En:

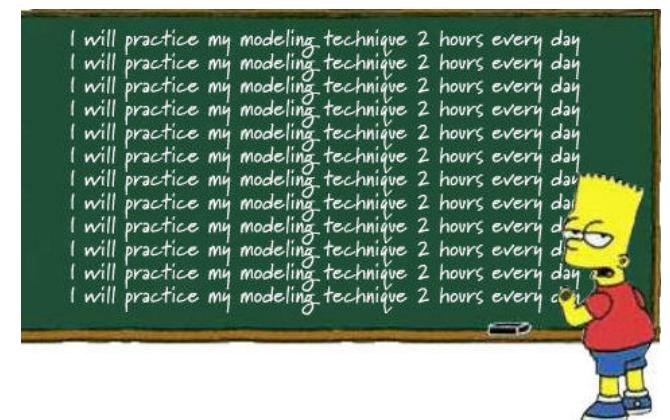
```
function init(e)
{
    var myText = document.getElementById('myText')
    addEvent(myText, 'keyup', presskey, false);
}

function presskey(e) {
    var key = e.keyCode;
    alert("Je hebt gedrukt op " + String.fromCharCode(key));
}
```

*Uitzondering: AngularJS. Hierin wél inline event handlers
(maar: andere omgeving/scope!)*

Oefening

- Implementeer het voorbeeld uit de vorige slide
 - Vang via unobtrusive JavaScript een toetsindruk af
 - toon in de console of op een element in de pagina op welke toets werd ingedrukt.

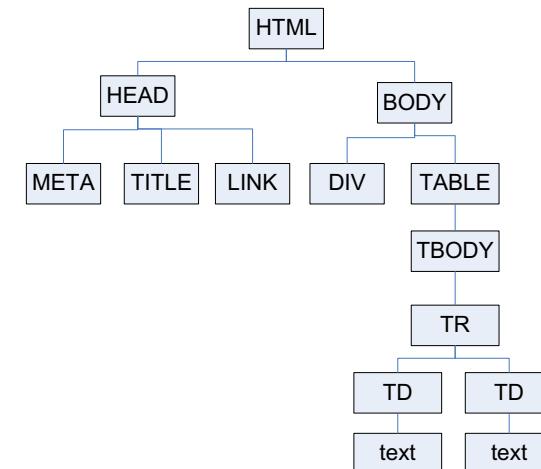


Event object

- De functie die wordt aangeroepen ontvangt een event object als parameter
- Belangrijke eigenschappen:
 - target (het element dat het event heeft afgevuurd)
 - type (het soort event)
 - button (welke muisknop ingedrukt was bij mouseevent - 0: left, 1: middle, 2: right)
 - keyCode
 - shiftKey

Events – meer kenmerken

- Bubbling:
 - het event 'reist' van het laagst gelegen element naar het hoogste
 - kun je stoppen met `stopPropagation()`;
- Default acties
 - de standaard actie die bij een bepaald event wordt uitgevoerd
 - kun je stoppen met `e.preventDefault()`;



Event - voorbeeld

```
8 <div id="container">
9 <button id="btnHello">Say Hello!</button>
10 </div>
11 <script>
12 var button = document.getElementById('btnHello');
13 button.addEventListener(
14     'click', // eerste argument
15     function(){ // tweede argument
16         alert('Hello!');
17     },
18     false); // derde argument
19 </script>
20
```

Timing: de event `window.onload()`

```
7 // Voorbeeld: tekstinvoer in tekstveld afvangen
8 window.onload=function(){
9     var txt = document.getElementById('txtName');
10    txt.addEventListener(
11        'keyup',                      // eerste argument
12        handleKey,                  // tweede argument: functieaanroep
13        false);                     // derde argument
14 }
15
16 // Event handler
17 function handleKey(e) {
18     var keyCode = e.which;
19     var alinea = document.getElementById('keyCodes');
20     alinea.innerHTML += ', ' + keyCode;
21 }
22
23</script>
24</head>
25<body>
26
27<div id="container">
28    <input type="text" id="txtName" />
29    <p>Code(s) van de toetsen: </p>
30    <p id="keyCodes"></p>
31</div>
32
```

Oefening

- Maak een script waarbij na een klik op de knop alle links in de pagina een andere achtergrondkleur krijgen.
- Gebruik:
 - Een array met kleuren
 - getElementsByTagName ()
 - Loop over een NodeList-object

