

CVO TKO Leuven

Clientside scripting

Javascript

Martijn Waeyenbergh



Javascript

Inhoud

Inleiding.....	4
Geschiedenis	4
Client-side	4
Lage drempel, veel mogelijkheden	5
Frameworks	5
Javascript in HTML	5
Script in de pagina.....	5
Script in een extern bestand	6
Script ondersteuning in browsers.....	7
Pop-ups	8
Info-popup (alert)	8
Confirm-popup (OK/Annuleer)	8
Prompt-popup (OK/Annuleer)	10
Javascript : aan de slag!	11
Elementen aanpassen via script	12
OPGELET: hoofdletters!	14
Rechtstreeks schrijven	14
Punt Komma.....	14
Een woordje uitleg... voor jezelf	15
Voorbeeld in de praktijk:	16
ID en class	17
Functies	18
Variabelen in javascript.....	20
To var, or not to var	20
Variabelen - Types - Waardes	21
Numerieke variabelen.....	21
Tekst variabelen (String)	21
Boolean variabelen	22
Reeks variabelen (Array)	22
Object variabelen	23
Globale en lokale variabelen.....	24
Operatoren.....	26

Beslissingsstructuren	27
If-else & if-else if-else.....	27
Switch.....	28
Lus structuren	29
De for herhalingslus	29
De for/in herhalingslus.....	30
De While loop.....	31
De Do/While loop	31
Break	31
Continue.....	32
Voorbeeld : Teken een kerstboom	32
Voorbeeld : Blad steen schaar	33
Arrays	35
Enkele veelgebruikte array methodes:	35
Lenght	35
indexOf.....	36
concat.....	36
Join	36
Pop	37
Push.....	37
Reverse.....	37
Shift	38
Slice	38
Sort.....	38
Splice	39
toString.....	40
Unshift.....	40
Voorbeeld : Adventure Game	41
Voorbeeld : Auto.....	48
Voorbeeld : Pacman	52
Voorbeeld: Puzzel	62
Timer functies	65
Timer events in Javascript.....	65
setInterval	65

setTimeout	65
clearTimeout	65
Voorbeeld : de stopwatch	66
Voorbeeld: Keukenwekker	68
Objecten	70
Properties en Methoden	70
Andere objecten : Math	70
Andere objecten : Date	72
Eigen Objecten maken	75
Properties gebruiken	76
Voorbeeld : feest	78
Jquery	82
JQuery en andere frameworks	82
jQuery	82
Versies	82
jQuery pakketen	83
jQuery in gebruik	83
CDN's	83
jQuery insluiten	84
jQuery shorthands	85
jQuery Selectors	86
jQuery functies	88

Inleiding

Geschiedenis

De eerste versie van Javascript werd in 1995 ontwikkeld door Brendan Eich van Netscape Communications Corporation voor gebruik in Netscape Navigator.

Aanvankelijk was de naam Mocha en vervolgens LiveScript. De taal werd hernoemd tot Javascript in de tijd dat in de Netscape-browser ook ondersteuning voor Java-applets werd ingebouwd.

Hoewel er veel verwarring over bestaat, heeft Javascript, buiten de naam; eigenlijk helemaal niets met de programmeertaal Java te maken.

Vandaag is Javascript de meest populaire programmeertaal ter wereld!



Client-side

Javascript is een scriptingtaal die op de lokale pc draait. Dit is een groot voordeel en maakt dat er geen gebruik moet gemaakt worden van een client-server architectuur. (geen server nodig!)

Client-side scripting wordt uitgevoerd in de lokale webbrowser. Deze zal de code interpreteren en uitvoeren.

Javascript wordt gebruikt binnen een HTML pagina (webpagina). Deze pagina wordt in de webbrowser getoond, en de code wordt geïnterpreteerd en uitgevoerd.

Hoewel javascript omschreven is in duidelijke regels en afspraken, zal je ongetwijfeld merken dat niet alle browsers deze regels even strikt naleven en dus afwijkende resultaten kunnen geven.

Deze manier wordt typisch gebruikt wanneer men (delen van) een webpagina dynamisch wil laten veranderen, tijdens of na het laden van de gehele pagina. Sinds het gebruik van AJAX zit de populariteit van javascript enorm in de lift. In Windows 8 is javascript tevens een volwaardige ontwikkeltaal voor het programmeren van applicaties.

Lage drempel, veel mogelijkheden

Javascript is relatief eenvoudig om aan te leren. Hoewel de taal uitgegroeid is tot een volwaardige programmeertaal. Voorbeelden zijn oa. Windows 8, web-toepassingen...

Frameworks

Dankzij frameworks wordt werken met javascript nog makkelijker en aangenamer. Deze frameworks zijn in essentie functies en patronen om bepaalde veelgebruikte acties te vereenvoudigen. De meest bekende van deze frameworks is ongetwijfeld JQuery. (<http://www.jquery.com>)

Javascript in HTML

Hoewel javascript een programmeertaal is die zelfstandig kan werken, wordt deze in de meeste gevallen gebruikt binnen een webpagina.

Webpagina's zijn opgebouwd in een bepaalde structuur of taal. Deze "taal", heet HTML. (Hyper Text Markup Language)

De structuur van HTML ligt eveneens, net als javascript, vastgelegd in gemaakte afspraken. De overkoepelende organisatie die deze afspraken opstelt en controleert is het W3C consortium. (<http://www.w3.org>)

Het script gedeelte van een pagina kan je ofwel in je pagina zelf zetten (tussen de script tags), ofwel apart in een afzonderlijk bestand (met .js als extensie).

Script in de pagina



In de meeste gevallen zal men de scripts in de <head> sectie van een html bestand plaatsen, maar dit is niet verplicht. Je kan een oneindig aantal scripts in een pagina plaatsen, op eender welke plaats.

Om alles overzichtelijk te houden, plaatst men deze meestal in de 'head' sectie, of helemaal onderaan de pagina.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       document.write("<h1>Hallo Iedereen</h1>");
6.     </script>
7.   </head>
8.   <body>
9.     Dit is mijn HTML pagina
10.  </body>
11. </html>
```

Wanneer je je javascript in een pagina wil zetten, moet deze tussen 2 'tags' gezet worden. Voor je script zet je de tag "<script>" en op het einde, als laatste, zet je "</script>".

Script in een extern bestand

In veel gevallen wil men de webpagina zo overzichtelijk mogelijk houden, en alle niet strikt nodige code eruit houden. Dit kan je perfect doen door je javascript bestand als een extern bestand te bewaren, en in je HTML pagina een verwijzing te leggen naar dit bestand. Een javascript bestand heeft als extensie altijd ".js", wat natuurlijk voor "javascript" staat.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script src="./js/mijnScript.js"></script>
5.   </head>
6.   <body>
7.     Dit is mijn HTML pagina
8.   </body>
9. </html>
```

In de regel zie je ook het attribuut 'src' staan, wat voor 'source' of bron staat. Dit is de plaats waar je het javascript kan terugvinden.

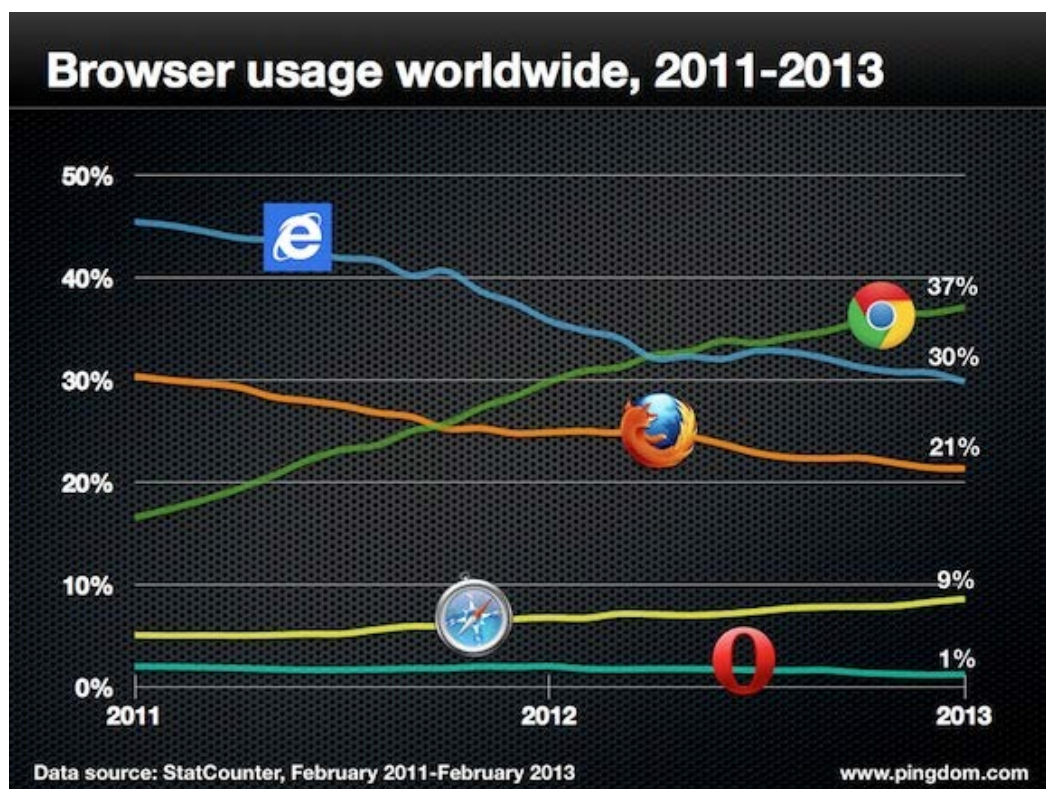
Een goed idee als je op zo'n manier werkt, is het logisch ordenen van je script bestanden in een aparte folder. Maak bijvoorbeeld een 'js' folder aan, waarin je je verschillende javascript bestanden bewaard.

Script ondersteuning in browsers

Over het algemeen is javascript op alle browsers toegankelijk, maar dat is natuurlijk niet altijd zeker. Javascript ondersteuning kan je manueel afzetten in je browser (voor eventuele veiligheidsoverwegingen), waardoor je script niet uitgevoerd zal worden. Om dit op te vangen, en een mogelijke 'waarschuwing' te geven aan de gebruiker, kan je gebruik maken van de <noscript> tag.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       document.write("<h1>Hallo Iedereen</h1>");
6.     </script>
7.     <noscript>
8.       Helaas, activeer javascript voor deze website!
9.     </noscript>
10.  </head>
11.  <body>
12.    Dit is mijn HTML pagina
13.  </body>
14. </html>
```

Deze tag zal weergegeven worden wanneer javascript niet actief is.



Pop-ups

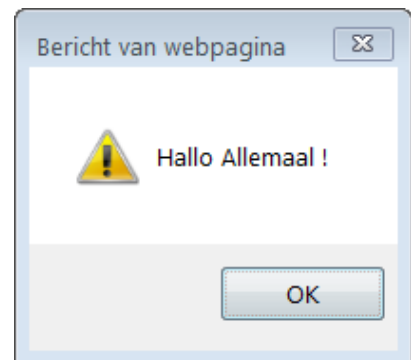
Niets zo vervelend als die popups, maar soms zeer handig!

Er bestaan in javascript een aantal mogelijkheden, zo kan je een popup tonen, enkel met informatie in, of je kan via een popup net informatie vragen...

Info-popup (alert)

```
1. window.alert("Hallo Allemaal!");
```

De meest eenvoudige van de popups is de informatie popup. Deze kan je oproepen door de functie 'alert' te gebruiken. De functie kan je ook oproepen zonder de "window" ervoor.



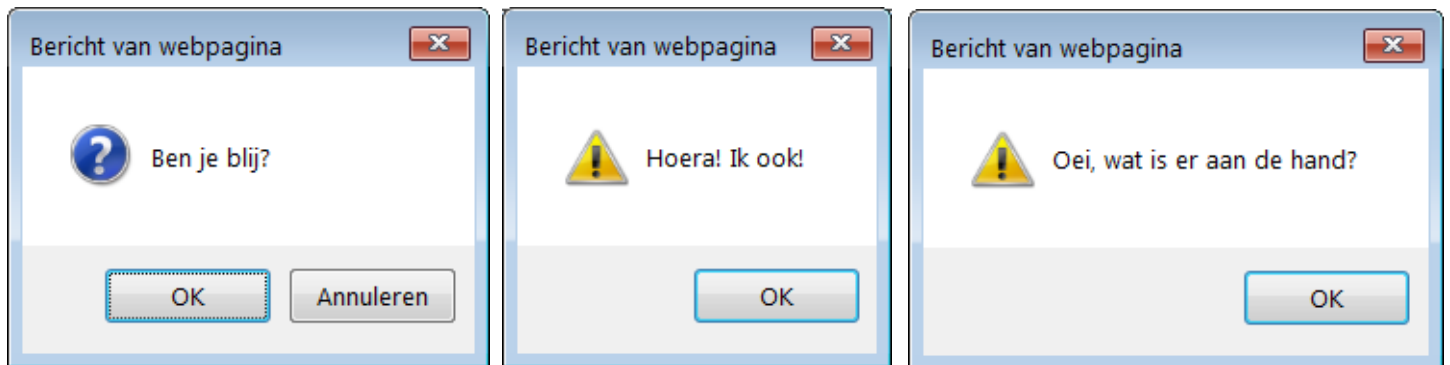
Confirm-popup (OK/Annuleer)

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       var r=confirm("Ben je blij?");
6.       if (r==true)
7.       {
8.         x="Hoera! Ik ook!";
9.       }
10.      else
11.      {
12.        x="Oei, wat is er aan de hand?";
13.      }
14.      alert(x);
15.    </script>
16.  </head>
17.  <body>
18.  </body>
19. </html>
```

Met deze popup kan je een vraag stellen waarop de gebruiker kan accepteren (OK) of kan afwijzen (Annuleer).

Het antwoord op deze, kan je via een if/else structuur nagaan en eventueel op reageren... (OK = true en Annuleer = False)

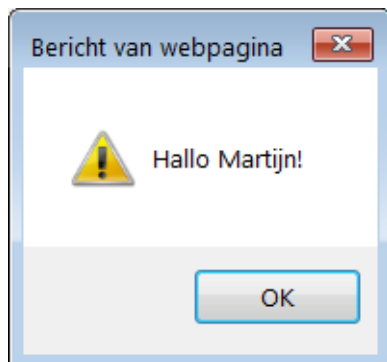
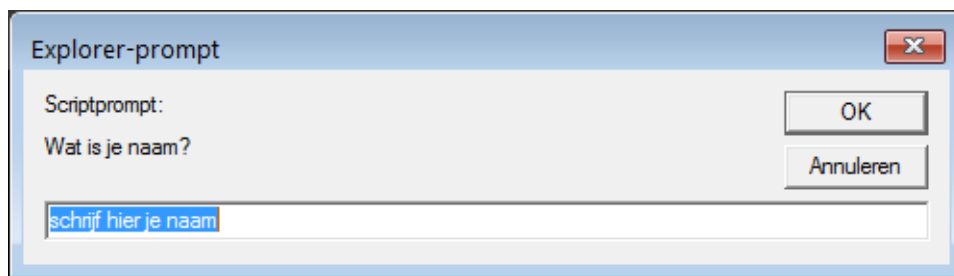
Deze popup wordt zeer veel gebruikt als "ben je zeker dat..." bij bijvoorbeeld het verwijderen van data...



Prompt-popup

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       var naam=prompt("Wat is je naam?","schrijf hier je naam");
6.       alert("Hallo " + naam + "!");
7.     </script>
8.   </head>
9.   <body>
10.  </body>
11. </html>
```

Met deze popup kan je een vraag stellen waarop de gebruiker niet alleen kan accepteren (OK) of kan afwijzen (Annuleer), maar ook naar invoer gevraagd wordt. (in het voorbeeld vragen we de naam).



Javascript : aan de slag!



Omdat je javascript script draait in je webpagina, en niet als een apart programma op je pc, kan dit script ook alleen werken ‘binnen’ de pagina. Het kan met andere woorden ook alleen maar elementen binnen je pagina gebruiken. Het is ook zo, dat javascript geen toegang kan krijgen tot je lokale pc (bijvoorbeeld het wegschrijven, of lezen van data naar/van een bestand).

Maar ondanks deze ‘restrictie’, kan javascript wel heel veel. Zo kan je alle elementen in je pagina gebruiken, aanpassen, verwijderen... en zelfs dynamisch nieuwe elementen toevoegen.

Als je een element wil gebruiken, moet je natuurlijk eerst kunnen specificeren welk element je juist wil gaan gebruiken. Dit kan je op verschillende manieren. De meest gebruikte manier, is het zoeken van een element, met een referentie naar de unieke ‘ID’.

```
var element = Document.getElementById("unieke_id");
```

Met deze regel laat je je script zoeken naar een element met een ID gelijk aan “unieke_id”. Een referentie naar dit element zal dan in de variabele “element” gezet worden, waarmee je dan makkelijk aan de slag kan.

In HTML kan je verschillende elementen op de pagina zetten. (bv. Een tekst in Hoofding 1 (<h1> een grote kop), een paragraaf (p), een link naar een andere pagina (< a href),...) Omdat er veel verschillende elementen op je pagina staan, is het makkelijk al deze elementen een unieke naam te geven. Dit is dan de unieke ID (identifier), waarmee je kan verwijzen naar dit element.

Vb:

```
1. <!DOCTYPE html>
2. <html>
3.   <head></head>
4.   <body>
5.     <h1 id="groteTitel">Hallo Iedereen</h1>
6.   </body>
7. </html>
```

Zoals je kan zien, moet je elk (met uitzondering van enkele) element openen en sluiten. Openen doe je door de tag te openen (<h1>) en sluiten doe je door deze opnieuw te herhalen, maar dan met een voorliggende schuine streep (</h1>).

Eveneens zie je de unieke id van dit element (id="groteTitel"). Dit kunnen we gebruiken om naar dit specifieke element te verwijzen.

Er bestaan nog veel meer attributen van de verschillende elementen, en niet alle elementen hebben dezelfde attributen... maar daar komen we tijdens de cursus nog geregeld op terug.

Even resumeren: We gebruiken tags om elementen te openen en te sluiten, hier kan je extra attributen definiëren (bv. ID)

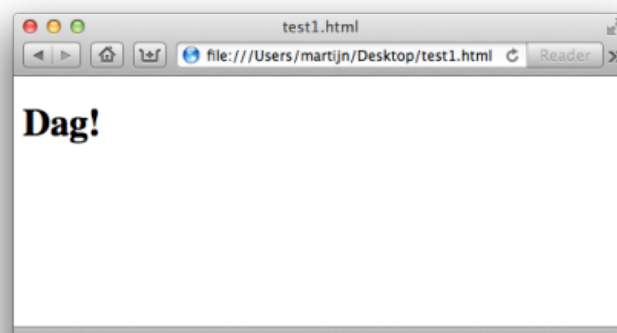
Elementen aanpassen via script

Als je een element wil wijzigen, moet je eerst specificeren over welk object het gaat. Eenmaal dit duidelijk is, kan je gaan specificeren welk attribuut je wil wijzigen.

Stel, we hebben een <h1> met een bepaalde tekst (zoals in het vorige voorbeeld), en je wil de tekst van dit element aanpassen.

```
1. <!DOCTYPE html>
2. <html>
3.
4.   <head>
5.     <script>
6.       window.onload = function(){
7.         document.getElementById("groteTitel").innerHTML = "Dag!";
8.       }
9.     </script>
10.  </head>
11.  <body>
12.    <h1 id="groteTitel">Hallo Iedereen</h1>
13.  </body>
14. </html>
```

Als je deze code als HTML bestand bewaart, en in een browser toont, zal je zien dat het resultaat er als volgt zal uitzien:



Wat betekent deze code nu eigenlijk juist?

```
1. window.onload = function(){};
```

Hier gaan we bepalen, dat het laden van de pagina een functie laat starten. Dit kan je zien als, “als de pagina geladen is, moet je hetgeen tussen de accolades staat uitvoeren”.

Een andere manier om dit te bereiken (wordt ook zeer veel gebruikt), is de functie starten door dit aan te geven in de <body> tag. Als je dit zo wil doen, ziet het er zo uit:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       function zegDag()
6.       {
7.         document.getElementById("groteTitel").innerHTML = "Dag!";
8.       }
9.     </script>
10.  </head>
11.  <body onload="zegDag()">
12.    <h1 id="groteTitel">Hallo Iedereen</h1>
13.  </body>
14. </html>
```

Zoals je hier kan zien, zal tijdens het laden de body tag gelezen worden, en zal de onload functie van de body uitgevoerd worden. Hierin staat een oproep naar een functie “zegDag”, waarin onze code staat.

```
1. document.getElementById("groteTitel").innerHTML = "Dag!";
```

Met deze code zullen we ons richten op een element met een bepaalde ID. Meerbepaald, het element met ID = “groteTitel”.

We vragen aan het document (dat is de pagina waarin we bezig zijn), te kijken naar een element met een id. Dat doen we door de methode “getElementById()” te gebruiken. getElementById geeft ons een verwijzing naar het element, waarmee we dan iets kunnen gaan doen. Zoals hier, gaan we het attribuut ‘innerHTML’ gebruiken en aanpassen.

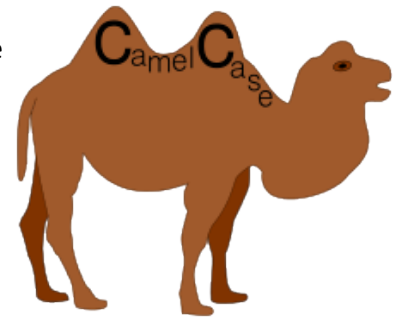
Dit kunnen we door er gewoon “.innerHTML” achter te typen. Dit betekent dat we deze eigenschap van dit element kunnen gebruiken. Door er een gelijkaan (=) teken achter te zetten, kunnen we een nieuwe waarde toekennen. (bijvoorbeeld “Dag!”).

Merk op dat het toewijzen van een nieuwe waarde gebeurt door een enkel gelijkaan teken!

OPGELET: hoofdletters!

Het hoofdlettergebruik in javascript is zeer belangrijk! Als er een hoofdletter moet staan, moet ze er ook staan, anders is het niet correct!

Het hoofdlettergebruik heetten we "CamelCase", met een knipoog naar de 2 bulten van een kameel. Met deze 'methode' gebruiken we bij elk nieuw woord, een hoofdletter. (met uitzondering van het eerste woord). Spaties worden nooit gebruikt!



Enkele voorbeelden: "groteTitel" of "ditIsEenLangeNaam".

Rechtstreeks schrijven

Als je rechtstreeks in de pagina wil schrijven, kan je werken via de functie "write". Deze functie laat je toe rechtstreeks iets te schrijven in de webpagina.

```
1. document.write("<p>Dit is een lijn tekst");
```

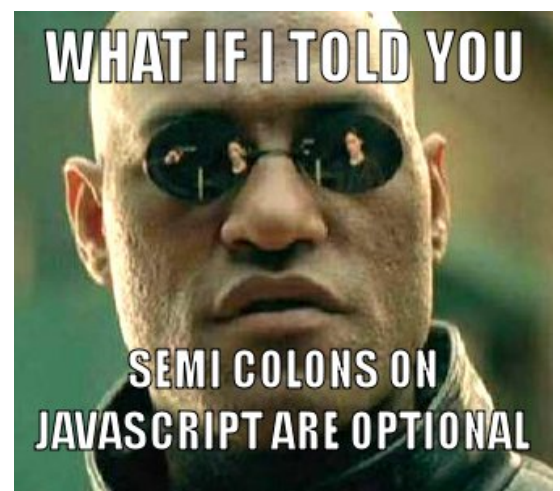
Door deze code zal er in je document een <p> element (een paragraaf) toegevoegd worden, met als inhoud "Dit is een lijn tekst".

OPGELET: als je deze functie gebruikt, zal de huidige inhoud van je pagina overschreven worden door deze nieuwe inhoud! Dat kan er dus voor zorgen dat plots al je inhoud weg is.

Punt Komma

Elke lijn javascript die je schrijft zou in principe moeten afgesloten worden met een puntkomma (;)

Dan weet de javascript engine dat deze lijn volledig is. Hoewel dit als 'best practise' geschreven wordt, is de puntkomma aan het einde van de regel geen verplichting.



Een woordje uitleg... voor jezelf

Commentaar.

Commentaar is belangrijk, niet alleen voor anderen, maar ook voor jezelf! Commentaar is tekst dat je kan toevoegen aan je code, zonder dat deze code geïnterpreteerd zal worden door de browser.

Dit laat je toe om een beetje meer uitleg (achtergrond info, meer uitleg over wat je daar doet, of bedenkingen (todo's)) aan je code toe te voegen.

Commentaar is ook zeer handig om bepaalde lijnen code even 'on-hold' te zetten, zonder ze kwijt te spelen. Als je je code aan het schrijven bent, zal je merken dat het goed is af en toe te kunnen testen (debuggen), om eventuele fouten eruit te halen. Dan is het handig om stukken even uit te schakelen, en ze later weer actief te maken.

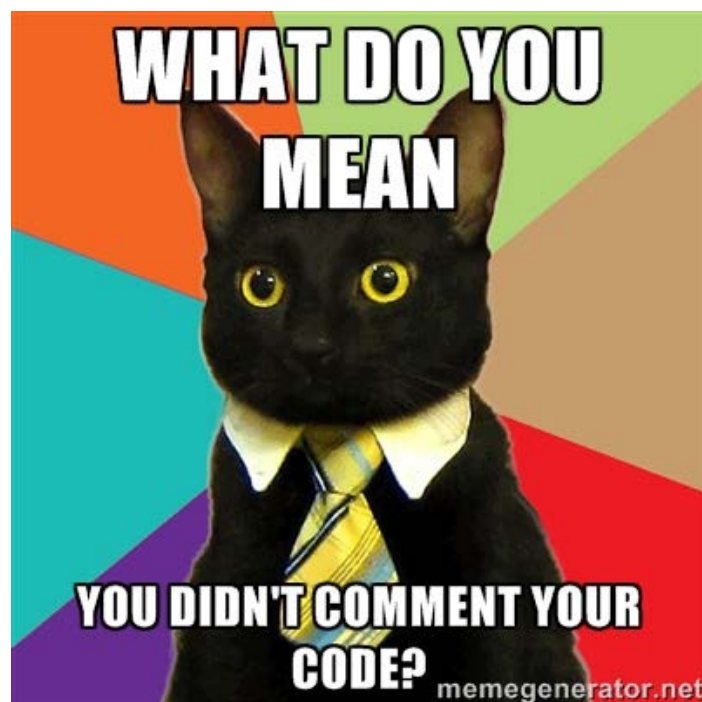
Commentaar in javascript is heel eenvoudig:

```
1. // dit is een lijn commentaar
```

1 lijn commentaar doe je door voor de lijn tekst `“//”` (een dubbele schuine streep) te plaatsen.

Als je in 1 keer meerdere lijnen code in commentaar wil zetten, kan je dan door vooraan `“/*”` te zetten, en achter de code die in commentaar moet komen, een `“*/”` te zetten. Alles wat tussen `/*` en `*/` staat, staat in commentaar.

```
1. /*  
2. dit zijn meerdere lijnen commentaar  
3. dit zijn meerdere lijnen commentaar  
4. */
```

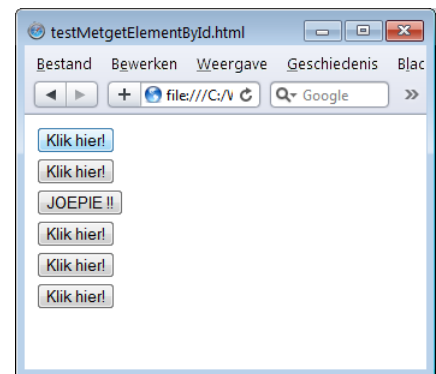


Voorbeeld in de praktijk:

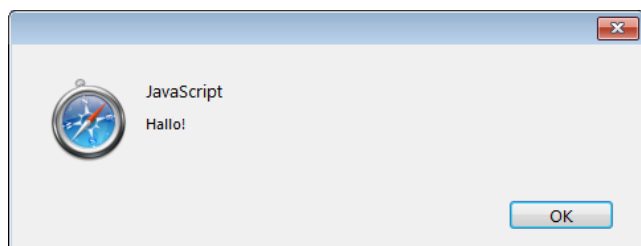
```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       function veranderTekst(){
6.         document.getElementById("knop3").innerHTML = "JOEPIE !!";
7.       }
8.     </script>
9.   </head>
10.  <body>
11.    <button id='knop1' class="knop" onclick="veranderTekst()">Klik hier!</button><br>
12.    <button id='knop2' class="knop" onclick="alert('Hallo!')">Klik hier!</button><br>
13.    <button id='knop3' class="knop" onclick="alert('Hallo!')">Klik hier!</button><br>
14.    <button id='knop4' class="knop" onclick="alert('Hallo!')">Klik hier!</button><br>
15.    <button id='knop5' class="knop" onclick="alert('Hallo!')">Klik hier!</button><br>
16.    <button id='knop6' class="knop" onclick="alert('Hallo!')">Klik hier!</button><br>
17.  </body>
18. </html>
```



→ klik op knop 1 →



→ Klik op eender welke knop,
behalve knop 1 →



ID en class

Wat is het verschil tussen een id en een class?

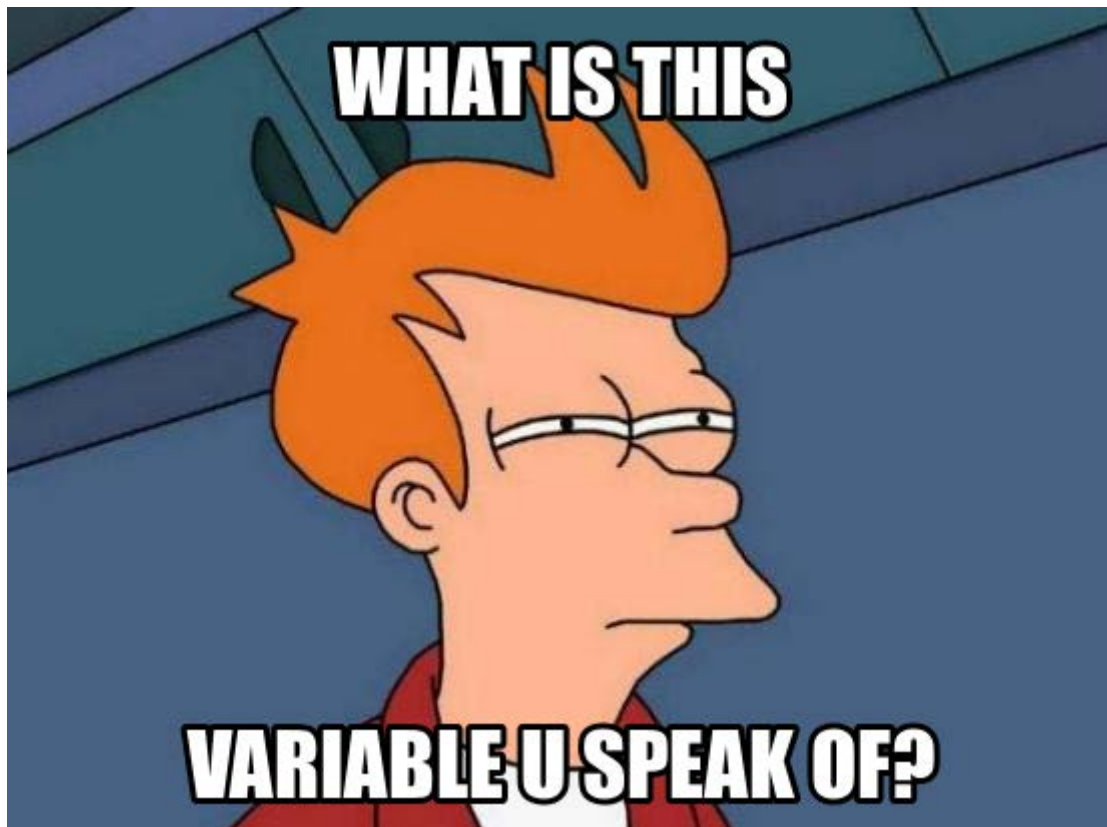
Id

een id is uniek, en kan maar 1 keer voorkomen op de pagina. Dat wil zeggen, dat dit het element uniek benoemt.

Class

Een klasse is een verzameling van 1 of meer elementen. Er kunnen meerdere elementen in dezelfde klasse zitten. Dit wordt zeer veel gebruikt voor het toepassen van stijlen en opmaak, maar ook in javascript.

Stel, je wil 1 knopje aanpassen, dan werkt je het makkelijkste met een verwijzing naar de correcte id. Als je anderzijds, alle knopjes van een bepaalde klasse wil aanpassen, kan je beter werken met een verwijzing naar de class.



Functies

Functies in javascript geven je de mogelijkheid een reeks opdrachten uit te voeren, wanneer je ernaar "vraagt".

Een functie bevat dus 1 of meerdere lijnen code die zullen uitgevoerd worden als je de functie aanroept.

Typisch ziet een functie er als volgt uit:

```
1. function naamVanDeFunctie(argument1, argument2){
2.     /*
3.         ... hier komt de code ...
4.     */
5.
6.     // indien de functie een waarde teruggeeft, komt er een return
7.     return terugGeefWaarde
8. }
```

Belangrijke samenvatting:

- Een functie, moet je starten door het codewoord "function".
- Achter de functienaam staan haakjes, waarbinnen eventuele argumenten komen.
- Een functie kan invoer-parameters vragen (argumenten)
- Indien er meer dan 1 invoer argument gegeven is, plaats je een komma tussen de verschillende waardes.
- Een functie kan een waarde teruggeven
- De code binnen een functie staat tussen twee accolades {}

De naam van je functie volgt dezelfde regels die gelden voor variabelen. Dat wil zeggen, dat je alle letters en cijfers mag gebruiken, inclusief underscore (_) en dollarteken (\$). Maar zeker geen spaties of andere speciale tekens!

Sommige functies worden uitgevoerd wanneer er een bepaalde actie gebeurde (bv. Een knop werd aangeklikt), of een functie kan manueel aangeroepen worden vanuit een lijn code in het script.

Stel: je wil een functie die twee getallen optelt:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title></title>
5.     <script>
6.       function optellen(){
7.         var getal1 = document.getElementById("getalInvoer1").value;
8.         var getal2 = document.getElementById("getalInvoer2").value;
9.         document.getElementById("resultaat").value = parseInt(getal1) + parseInt(getal2);
10.      }
11.    </script>
12.  </head>
13.  <body>
14.    <h1>Optellen</h1>
15.
16.    Getal 1 : <input type="text" value="" id="getalInvoer1"><br>
17.    Getal 2 : <input type="text" value="" id="getalInvoer2"><br>
18.    <input type="button" value="Tel Op !" id="optelKnop" onClick="optellen();"><br>
19.    Uitkomst : <input type="text" value="" id="resultaat"><br>
20.
21.  </body>
22. </html>
```

Een ander voorbeeld:

Een pagina met bovenaan een kader, waarin je tekstlijnen kan toevoegen via een invulveld en een knop. Als je tekst invult, en op de knop klikt, zal de tekst opgenomen worden in het bovenstaande vlak.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script type="text/javascript">
5.       function voegToe(){
6.         var lijst = document.getElementById("lijst").innerHTML;
7.         var nieuweWaarde = document.getElementById("invoerVak").value;
8.         lijst += nieuweWaarde + "<br>";
9.         document.getElementById("lijst").innerHTML = lijst;
10.        document.getElementById("invoerVak").value = "";
11.      }
12.    </script>
13.
14.  </head>
15.  <body>
16.    <div id="container">
17.      <div id="lijst"></div>
18.      <hr>
19.      <div id="invoer">
20.        <input type="text" value="" id="invoerVak">
21.        <button id="voegToeKnop" onclick="voegToe()">Voeg Toe</button>
22.      </div>
23.
24.    </div>
25.  </body>
26. </html>
```

Variabelen in javascript

Variabelen zijn '**containers**' waarin je informatie kan bewaren.

Enkele belangrijke afspraken over naamgeving van een variabele:

- Variabelnamen moeten beginnen met een letter (!)
- Variabelnamen mogen ook beginnen met \$ of _
- Variabelnamen zijn hoofdlettergevoelig (a \neq A)
- Je mag gebruik maken van alle letters en cijfers (a-Z 0-9)

To var, or not to var



Variabelen kunnen lange of korte namen hebben, zolang het voor jezelf duidelijk is wat ermee bedoeld wordt, is het goed.

Bij het maken van een nieuwe variabele, zal je deze eerst 'declareren' (aanmaken) met het codewoord 'var'

Vb: `var mijnVariabele;`

Op deze regel declareerde je een nieuwe variabele `mijnVariabele`. Deze variabele is nu gekend in het script en kan gebruikt worden. Hoewel deze momenteel nog geen waarde bevat (undefined)

Als je scripts zal doornemen, zal het je opvallen dat niet altijd alle variabelen voorafgegaan worden door het keyword 'var'. Dit is inderdaad niet verplicht, maar wel zeer aangewezen! Er zit namelijk een verschil in betekenis (later meer hierover).

Typisch zal je een variabele ook direct een waarde geven:

```
1. var mijnVariabele = 42;
```

Indien je meerder variabelen ineens wil aanmaken, kan je deze samen declareren en waardes toekennen:

```
1. var mijnVar1 = 42, mijnVar2 = "hallo", mijnVar3 = "Volvo";
```

Variabelen - Types - Waardes

In javascript zijn er een aantal soorten van variabelen types.

Deze kunnen verschillende soorten van data bevatten.

- Numeriek (alles wat nummers en getallen zijn)
- Tekst (alles wat tekst of letters zijn)
- Boolean (Ja/Nee of Waar/Vals True/False)
- Reeks (Array = een lijst van waardes)
- Object (later meer)
- Null en Undefined

undefined → wanneer er nog geen waarde aan de variabele toegekend is.

null → de variabele is niet gedefinieerd (en zal ook verwijderd worden uit geheugen)

Numerieke variabelen

In tegenstelling tot vele programmeertalen waar er veel verschillende soorten van getallen bestaan, bestaat er in javascript maar 1 definitie voor een getal.

Nummers kunnen geschreven worden met of zonder decimaal teken (komma getal)

```
1. var getal1 = 42;  
2. var getal2 = 42.005;
```

Tekst variabelen (String)

Tekst (Strings) kunnen alle tekst zijn, als ze tussen aanhaalingstekens staan! (enkele of dubbele)

OPGELET: als je tekst variabelen gebruikt, kan je kiezen welke aanhaalingstekens je gebruikt, maar je mag ze niet wisselen!

```
1. var tekst1 = "Dit is een tekst";  
2. var tekst2 = 'en dit ook';
```

Wat wel kan en mag, is enkele in dubbele gebruiken en omgekeerd.

```
1. var tekst1 = "Dit is een tekst"; // DIT IS FOUT !!  
2. var tekst2 = 'dit is correct';  
3.  
4. var tekst3 = "dit is een tekst met 'aanhaalingstekens';  
5. var tekst4 = 'dit is een tekst met "aanhaalingstekens";
```

opgelet wanneer je tekst en getallen mixed:

```
1. var getal = "5" + 5; // dit zal als resultaat 55 geven  
2. var getal = 5 + "5"; // dit zal als resultaat 55 geven  
3. var getal = 5+5 + "5"; // dit zal als resultaat 105 geven  
4. var getal = "5" + 5 + 5 ; // dit zal als resultaat 555 geven
```

Boolean variabelen

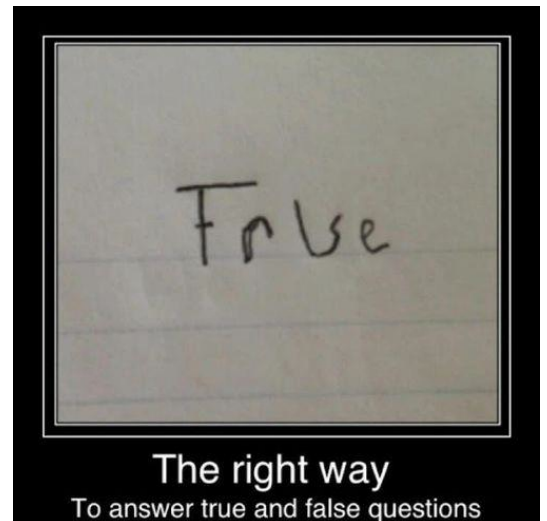
Boolean variabelen zijn zeer eenvoudige variabelen: ze hebben slechts 2 mogelijke waardes = "true" of "False".

True en False zijn geen tekst! Dit is een waarde waarop je kan testen!

```
1. var bool1 = true;  
2. var bool2 = false;
```

Booleans worden veel gebruikt als teruggave variabele van een functie.

Bedenk bijvoorbeeld dat je een functie hebt die nakijkt of het spel begonnen is (isGameStarted ()), dewelke dan true of false teruggeeft.

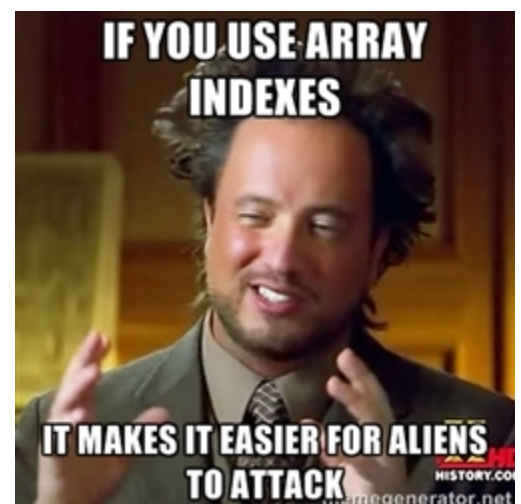


Reeks variabelen (Array)

Reeksen of Arrays zijn een verzameling van elementen.

```
1. var automerken=new Array();  
2. automerken [0]="Saab";  
3. automerken [1]="Volvo";  
4. automerken [2]="BMW";  
5.  
6. // dit kan ook op volgende manier:  
7. var automerken= new Array("Saab", "Volvo", "BMW");  
8.  
9. // of zelfs zo:  
10. var automerken= ["Saab", "Volvo", "BMW"];
```

Arrays zijn "zero-based", wat wil zeggen dat het eerste element uit de reeks als index getal 0 heeft, het 2de is dan 1, het 3de is 2... Later zien we nog veel meer mogelijkheden en gebruiken van arrays...



Object variabelen

In javascript is eigenlijk bijna alles een object. Objecten zijn elementen met eigenschappen (properties) en methodes.

Stel, je maakt een object "persoon". Dit object heeft een aantal eigenschappen (bv. Een naam, een lengte, een leeftijd, kleur van de ogen...)

```
1. function persoon(naam, lengte, leeftijd, oogKleur)
2. {
3.   this.naam=naam;
4.   this.lengte=lengte;
5.   this.leeftijd=leeftijd;
6.   this.oogKleur=oogKleur;
7. }
```

Wanneer we een eigenschap van een object willen gebruiken, kan je dat door de naam van het object.eigenschap te schrijven.

In ons voorbeeld van de persoon; als je de naam wil gebruiken:

```
1. persoon.naam=naam;
```

Je kan in een object ook zelf functies toevoegen. Bijvoorbeeld een functie die de naam van de persoon kan veranderen:

```
1. function persoon(naam, lengte, leeftijd, oogKleur)
2. {
3.   this.naam=naam;
4.   this.lengte=lengte;
5.   this.leeftijd=leeftijd;
6.   this.oogKleur=oogKleur;
7.
8.   this.veranderNaam = veranderNaam;
9.   function veranderNaam(nieuweNaam){
10.     this.naam = nieuweNaam
11.   }
12. }
```

Nu kan je dit als volgt gebruiken:

```
1. Persoon.veranderNaam("martijn");
```

Verder in de cursus gaan we nog veel verschillende eigen objecten aanmaken en verder uitdiepen... Omdat eigenlijk bijna alles een object is, kan je van bijna alles de eigenschappen en de methodes gebruiken. Enkele voorbeelden:

```
1. var naam = "Javascript";
2. document.write ("de lengte van deze tekst: " + naam.length);
```


Globale en lokale variabelen

Wanneer je variabelen gaat aanmaken binnen je script kan je dat op verschillende plaatsen. Dit is belangrijk omdat dit het "bereik" (scope) van de variabele bepaald.

Stel:

Je hebt 4 knopjes...

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.
6.       var begroeting = "Hallo, "; // dit is een globale variabele
7.
8.       function zegHallo1(naam){
9.         alert(begroeting + naam);
10.      }
11.
12.      function zegHallo2(naam){
13.        var begroeting = "Goeiedag, "; // dit is een lokale variabele
14.        alert(begroeting + naam);
15.      }
16.
17.      function zegHallo3(naam){
18.        begroeting = "Saluut, "; // dit is een globale variabele
19.        alert(begroeting + naam);
20.      }
21.
22.
23.
24.     </script>
25.   </head>
26.   <body>
27.     <button id='knop1' class="knop" onclick="zegHallo1('martijn')">Globaal</button><br>
28.     <button id='knop2' class="knop" onclick="zegHallo2('martijn')">Lokaal</button><br>
29.     <button id='knop3' class="knop" onclick="zegHallo3('martijn')">Globaal</button><br>
30.     <button id='knop1' class="knop" onclick="zegHallo1('martijn')">Globale -
31. > gewijzigd</button><br>
32.   </body>
33. </html>
```

1. Bij het klikken op de eerste knop, zal de globale variabele "begroeting" gebruikt worden.
2. Bij het klikken op de tweede knop, zal het script zijn eigen lokale variabele gebruiken.
3. Bij het klikken op de derde knop (**geen var**) zal het script de globale variabele gebruiken, en daarvan ook de waarde aanpassen!
4. Als resultaat van het klikken op de 3^{de} knop, zal nu ook het klikken op de 4^{de} knop hetzelfde resultaat geven.

Samenvatting:

- Als je een variabele maakt buiten een functie, is dit een globale variabele (te gebruiken in alle code)
- Als je een variabele maakt in een functie, met het code woordje 'var' ervoor, is dit een lokale variabele (enkel gekend binnen de functie, en ook alleen daar te gebruiken!)
- Als je het codewoord 'var' gebruikt bij het maken van een variabele binnen een functie, zal dit toch een globale variabele zijn! Als deze al zou bestaan, pas je dus eigenlijk de variabele aan (je wijzigt de waarde van de globale variabele)

```
1  
2  
3 var global = 10;  
4  
5 function fun() {  
6  
7     var local = 5;  
8  
9 }  
10  
11  
12  
13
```

global variable

local variable

Operatoren

Wanneer je met waarde en/of variabelen wil werken, moet je bepaalde bewerkingen kunnen doen. Deze bewerkingen kan je door middel van operatoren.

Plus, min, maal, gedeeld door... zijn operatoren

In javascript schrijf je deze zo:

+	(plus)	<code>var z = x + y;</code>
-	(min)	<code>var z = x - y;</code>
*	(maal)	<code>var z = x * y;</code>
/	(gedeeld door)	<code>var z = x / y;</code>
%	(restwaarde bij deling)	<code>var z = x % y;</code>
++	(plus 1)	<code>z++;</code>
--	(min 1)	<code>z--;</code>

Met een gelijkaan (=) wijs je een waarde toe aan een variabele

```
1. var naam = "Javascript";
```

De waarde "javascript" zal in de variabele naam bewaard worden.

Wanneer je een waarde bij de bestaande inhoud wil toevoegen, kan je werken met "+="

```
2. var naam = "Javascript";  
3. naam += " is actief!";
```

De waarde van de variabele "naam" zal na deze code dit zijn: **"Javascript is actief"**

Dezelfde logica gaat op voor volgende operatoren:

+=

-=

*=

/=

%=

Wanneer je 2 stukken tekst bij elkaar wil voegen, kan je werken met een plus teken (+).

```
1. var naam = "Javascript";  
2. document.write("<h1>" + naam + " is actief op deze computer. </h1><br>Hoera!!");
```

Zoals je kan zien, kan je met + meerder stukken aan elkaar "plakken" (concatenatie).

Zoals je misschien ook merkte, kan je in de uitvoer van je javascript **html** code toevoegen.

Controle structuren

Beslissingsstructuren

If-else & if-else if-else

De if - else structuur is een zeer veel gebruikte beslissingsstructuur in vele programmeertalen.

Aan de hand van een vraagstelling (een conditie) kan je een bepaalde blok code laten uitvoeren, of een andere. Afhankelijk van het antwoord zal het programma dus bepaalde code starten of overslagen.

De typische structuur voor deze beslissingsstructuur:

```
1. if (voorwaarde){  
2.     /*hier komt de code indien waar*/  
3. }else{  
4.     /*Hier komt de code indien nietwaar */  
5. }
```

Dit kan je bijna lezen. Je leest dan "Als de voorwaarde waar is, dan doe ik dit... indien niet, dan doe ik de code tussen de else accolades.

Een else is niet strikt noodzakelijk om deze structuur te gebruiken.

Een uitbreiding op deze structuur is, als je de else uitbreidt met een nieuwe if structuur. Dan krijg je dus een

```
If (voorwaarde){  
  
}else if( nieuwe voorwaarde){  
  
}else{  
  
}
```

In het volgende voorbeeld, gebruiken we de structuur voor een gepaste aanspreking te maken.

We maken een Datum object aan, en vragen daar de uren van op.

Indien het uur dat we hiervan terugkrijgen (dat is dus het huidige uur), kleiner is dan 12 weten we dat het nog voormiddag is, en zeggen we "Goeiemorgen!". Indien het uur groter is, maar kleiner dan de volgende voorwaarde zal het systeem "Goeiedag" zeggen... enzo verder...

Het voorbeeld in code:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title></title>
5.     <script>
6.
7.       var d = new Date();
8.       var n = d.getHours();
9.
10.      if (n < 12){
11.        document.write("Goeiemorgen !");
12.      }else if(n < 17){
13.        document.write("Goeiedag !");
14.      }else if(n < 23){
15.        document.write("Goeieavond !");
16.      }else{
17.        document.write("Goeienacht !");
18.      }
19.
20.    </script>
21.  </head>
22.  <body>
23.  </body>
24. </html>
```

Switch

De switch structuur kan men gebruiken wanneer er keuze gemaakt moet worden tussen verschillende mogelijke waarden. Elke waarde geeft je de mogelijkheid om andere acties uit te voeren.

Typisch ziet de structuur er zo uit:

```
1. switch(n)
2. {
3.   case 1:
4.     // voer deze code uit
5.     break;
6.   case 2:
7.     // voer deze code uit
8.     break;
9.   default:
10.    // voer deze indien de case niet in bovenstaande voorkwam...
11. }
```

Bovenaan de controlestructuur wordt een expressie geëvalueerd. In het bovenstaande voorbeeld "n". Deze waarde (dis is meestal een variabele) wordt dan per case afgetoetst.

Indien deze gevonden wordt, zal de code in deze case uitgevoerd worden. Deze case wordt dan afgesloten met het codewoord "break;".

Break zorgt ervoor dat de uitvoering stopt in de switch structuur.

Nog een voorbeeldje:

In dit voorbeeld vragen we de dag van de huidige datum. De waarde hiervan komt dan in de variabele dag te staan. Dan gaan we per case statement, deze waarde vergelijken met een nummer.

Indien gevonden, wordt er een nieuwe variabele 'x' gevuld met een tekst "Vandaag is het...".

```
1. var dag=new Date().getDay();
2. switch (dag)
3. {
4.   case 0:
5.     x="Vandaag is het zondag !";
6.     break;
7.   case 1:
8.     x="Vandaag is het maandag";
9.     break;
10.  case 2:
11.    x="Vandaag is het dinsdag";
12.    break;
13.  case 3:
14.    x="Vandaag is het woensdag";
15.    break;
16.  case 4:
17.    x="Vandaag is het donderdag";
18.    break;
19.  case 5:
20.    x="Vandaag is het vrijdag";
21.    break;
22.  case 6:
23.    x="Vandaag is het zaterdag";
24.    break;
25. }
```

Lus structuren

Lus structuren geven de mogelijkheid om bepaalde stukken code 1 of meerdere keren uit te voeren.

Er bestaan een aantal mogelijke lussen: de for-loop, de while en de do-while loop.

De for herhalingslus

De for loop is een van de meest gebruikte lussen. Dankzij een eenvoudige structuur, kan je met de for loop makkelijk code gaan herhalen.

```
1. for (var i=0; i<5; i++)
2. {
3.   document.write ("teller : " + i) ;
4. }
```

De for lus bestaat uit een aanroep van de "for" met daarin 3 argumenten.

Eerst definieer je een variabele voordat de loop start (var i = 0)

Dan definieer je de conditie wanneer de loop moet uitgevoerd worden. (i<5)

Dan zeg je wat er moet gebeuren per cyclus van de loop (i++) = tel 1 op bij de variabele i.

Argument 1 en 3 zijn optioneel binnen de for lus. Je kan deze buiten de lus definiëren:

Stel je schrijft het volgende:

```
1. var i = 0;
2. for (; i<5; i++)
3. {
4.     document.write("teller : " + i );
5. }
```

Of zelfs zo:

```
1. var i = 0;
2. for (;i<5;)
3. {
4.     document.write( "teller : " + i );
5.     i++;
6. }
```

Bovenstaande mogelijkheden zijn prima. Hoewel er meestal wel gewerkt wordt met de volledige 3 parameters in de for lus.

De for/in herhalingslus

De for-in loop wordt voornamelijk gebruikt om te lussen over de eigenschappen van een object.

```
1. var persoon={voornaam:"Jan",achternaam:"Janssens",leeftijd:25};
2.
3. for (x in persoon)
4. {
5.     tekst=tekst+ persoon[x];
6. }
7.
8. document.write tekst;
```

We maken hier een object "persoon" met een aantal eigenschappen. (voornaam, achternaam en leeftijd).

Met de hulp van de for loop kunnen we lussen over alle eigenschappen van het object.

De While loop

De while loop zal een blok code uitvoeren, zolang een gegeven voorwaarde waar is.

Bij het maken van een while loop geef je tussen de haken een conditie mee, waarop getest moet worden. Wanneer deze “waar” is, zal de code uitgevoerd worden. Opgelet voor oneindige lussen!

```
1. while (i<5)
2. {
3.     document.write ("teller : " + i);
4.     i++;
5. }
```

Bij elke uitvoering van de lus, zal de tekst uitgeschreven worden, en vervolgens de teller i met 1 verhoogd worden. Voor de volgende iteratie van de lus start, zal er eerst getest worden of deze conditie (i<5) nog waar is.

De Do/While loop

De do while loop is een variatie op de gewone while loop. Het verschilt van deze vorige doordat je eerst uitvoert, en dan pas test op de conditie. (dus er is zowieso een uitvoering van de code)

```
1. do
2. {
3.     document.write ("teller : " + i);
4.     i++;
5. }while (i<5);
```

Break

Het break woord, geeft aan dat je de lus wenst te doorbreken, en deze dus stopt.

Wanneer je dit gebruikt in een lus, zal de lus onderbroken worden, en verder gaan met de code na de lus...

```
1. for (i=0;i<10;i++)
2. {
3.     if (i==3)
4.     {
5.         break;
6.     }
7.     document.write( "teller " + i );
8. }
```

Dit zal als uitvoer hebben:

```
teller 0
teller 1
teller 2
```


Continue

Continue geeft aan dat je een bepaalde stap in de lus wil overslaan, maar de lus op zich nog wel verder gezet zal worden.

```
1. for (i=0;i<5;i++)
2. {
3.     if (i==3)
4.     {
5.         continue;
6.     }
7.     document.write( "teller " + i );
8. }
```

Dit zal als uitvoer hebben:

```
teller 0
teller 1
teller 2
teller 4
```

Voorbeeld : Teken een kerstboom

```
1. <!--
2.  *
3.  **
4.  ***
5.  ****
6.  *****
7.  *****
8.  -->
9.
10. <!DOCTYPE html>
11. <html>
12.     <head>
13.         <title></title>
14.         <script>
15.
16.             var uitvoer = "<center>";
17.             var sterretjes = "";
18.
19.             for(var teller = 0; teller < 20; teller++){
20.                 sterretjes = sterretjes + "*";
21.                 uitvoer = uitvoer + sterretjes + "<br>";
22.
23.             }
24.             uitvoer += "</center>";
25.             document.write(uitvoer);
26.
27.         </script>
28.     </head>
29.     <body></body>
30. </html>
```

Voorbeeld : Blad steen schaar

Een voorbeeld waarin we een werken met oa. Een functie, een if/els structuur.

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title></title>
5.
6.  <style>
7.  body{
8.      background-image: url("rock_paper_scissors_w1.jpeg");
9.      background-size: cover;
10. }
11. </style>
12.
13.     <script>
14.
15.         var SpelTeller = 0, aantalGewonnen = 0, aantalVerloren = 0, aantalGelijkSpel = 0;
16.
17.         function speel(inzet){
18.             SpelTeller++;
19.             var computerInzet = (Math.floor((Math.random()*3)+1));
20.             /* 1 = blad
21.                2 = steen
22.                3 = schaar
23.             */
24.             var result;
25.
26.             if (inzet == computerInzet){
27.                 result = "<span style='background-color:orange;'>GELIJKSPEL</span>";
28.                 ++aantalGelijkSpel;
29.             }else{
30.                 if (inzet == 1){
31.                     if (computerInzet == 2){
32.                         result = "<span style='background-
33. color:green; color:white;'>blad omwikkeld steen, ik win</span>";
34.                         ++aantalGewonnen;
35.                     }else{
36.                         result = "<span style='background-
37. color:red; color:white;'>schaar knipt door blad, de computer wint</span>";
38.                         ++aantalVerloren;
39.                     }
40.                 }
41.                 if (inzet == 2){
42.                     if (computerInzet == 3){
43.                         result = "<span style='background-
44. color:green; color:white;'>schaar breekt op steen, ik win</span>";
45.                         ++aantalGewonnen;
46.                     }else{
47.                         result = "<span style='background-
48. color:red; color:white;'>blad omwikkeld steen, de computer wint</span>";
49.                         ++aantalVerloren;
50.                     }
51.                 }
52.                 if (inzet == 3){
53.                     if (computerInzet == 1){
54.                         result = "<span style='background-
55. color:green; color:white;'>schaar knipt door blad, ik win</span>";
56.                         ++aantalGewonnen;
57.                     }else{
58.                         result = "<span style='background-
59. color:red; color:white;'>blad omwikkeld steen, de computer wint</span>";
60.                         ++aantalVerloren;
61.                     }
62.                 }
63.             }
64.         }
65.     }
66. </script>
67. </html>
```

```

53.         result = "<span style='background-
color:red; color:white;'>schaar breekt op steen, de computer wint</span>";
54.         ++aantalVerloren;
55.     }
56. }
57. }
58.
59.     var computerInzetTxt = "";
60.     computerInzetTxt = (computerInzet == 1 ) ? "blad" : "";
61.     computerInzetTxt = (computerInzet == 2 ) ? "steen" : "";
62.     computerInzetTxt = (computerInzet == 3 ) ? "schaar": "";
63.
64.     //alert ("computer speelde " +computerInzetTxt + "\n" + result);
65.     var scorebord = document.getElementById("scorebord");
66.     scorebord.innerHTML = "computer speelde " +computerInzetTxt + "<br>" + result
+ "<hr>";
67.     scorebord.innerHTML += "aantal gespeeld: " +SpelTeller + "<br>";
68.     scorebord.innerHTML += "aantal gewonnen: " +aantalGewonnen + "<br>";
69.     scorebord.innerHTML += "aantal verloren: " +aantalVerloren + "<br>";
70.     scorebord.innerHTML += "aantal gelijkspel: " + aantalGelijkSpel + "<br>";
71.
72.     if (SpelTeller >= 50){
73.         alert("ik denk dat je de oefening wel begrepen hebt, niet :)");
74.     }
75.
76.     /*
77.         MIJN      COMPUTER
78.         1          2 --> win ik
79.         1          3 --> computer wint
80.         2          1 --> computer wint
81.         2          3 --> win ik
82.         3          1 --> win ik
83.         3          2 --> computer wint
84.
85.     */
86.
87.
88.     }
89.
90.     </script>
91. </head>
92. <body>
93.
94.     <button onClick="speel(1)"><br>BLAD</button>
95.     <button onClick="speel(2)"><br>STEEN</button>
96.     <button onClick="speel(3)"><br>SCHAAR</button>
97.     <div id="scorebord">
98.         <!--Hier komt de score te staan...-->
99.     </div>
100.
101. </body>
102.</html>

```



Arrays

Reeksen of Arrays zijn een verzameling van elementen.

```
1. var automerken=new Array();
2. automerken [0]="Saab";
3. automerken [1]="Volvo";
4. automerken [2]="BMW";
5.
6. // dit kan ook op volgende manier:
7. var automerken= new Array("Saab","Volvo","BMW");
8.
9. // of zelfs zo:
10. var automerken= ["Saab","Volvo","BMW"];
```

Arrays zijn "zero-based", wat wil zeggen dat het eerste element uit de reeks als index getal 0 heeft, het 2de is dan 1, het 3de is 2.

Wanneer je een element uit een array wil aanspreken, kan je dat door de 'sleutel' van dit element te geven.

Bv. Geef het 2^{de} element uit de tabel:

```
1. var automerk = automerken[1];
```



Een array kan alle mogelijke objecten bevatten. Omdat eigenlijk alles in javascript een object is, betekent dit dat je er eigenlijk alles in kwijt kan. (getallen, teksten, arrays zelf, functies...)

Enkele veelgebruikte array methodes:

Lenght

De functie length geeft de lengte (het aantal elementen in de array) terug.

```
1. // maak array aan met 3 automerken in
2. var automerken= new Array("Saab","Volvo","BMW");
3.
4. // geef de lengte van deze array
5. document.write ("aantal elementen: " + automerken.length);
```

indexOf

indexOf geeft de positie van het gezochte element in de array.

OPGELET: deze functie werkt niet in browsers die javascript 1.6 niet gebruiken (bv. < IE8)

```
1. // maak array aan met 3 automerken in
2. var automerken= new Array("Saab","Volvo","BMW");
3.
4. // geef de positie van 'volvo'
5. document.write ("positie van 'volvo' : " + automerken.indexOf("Volvo"));
```

concat

met de concat functie kan je meerdere arrays samenvoegen tot 1.

```
1. // maak 3 arrays aan met
2. var studenten = ["Jan", "Piet"];
3. var erasmus = ["Joris", "Korneel", "Sandra"];
4. var vrijeStudenten = ["marcel", "Judith"];
5.
6. // voeg de 3 arrays nu samen tot nieuwe 1 array
7. var alleStudenten = studenten.concat(erasmus,vrijeStudenten);
8.
9. /*
10. het resultaat zal zijn:
11. --> Jan, Piet, Joris, Korneel, Sandra, marcel, Judith
12. */
```

Join

Met join kan je de inhoud van je array samenvoegen tot een string

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet"];
3.
4. // voeg de array nu samen tot nieuwe 1 tekst
5. document.write(studenten.join());
6.
7. /*
8. het resultaat zal zijn:
9. --> Jan, Piet
10. */
```

Pop

De pop methode, zal het laatste element uit de array teruggeven, en dit tevens ook verwijderen uit de array. De teruggeve waarde is het verwijderde element.

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris", "Korneel"];
3.
4. // geef laatste element terug en verwijder dit uit de array
5. document.write( "het laatste element: " + studenten.pop());
6.
7. /*
8.  het resultaat zal zijn:
9.  --> Korneel
10. de inhoud van de array na pop:
11. Jan, Piet, Joris
12. */
```

Push

De Push methode, zal een nieuw element aan de array toevoegen. (achteraan de array)

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris"];
3.
4. // voeg nieuw element toe aan de array
5. document.write(studenten.push("Korneel"));
6.
7. /*
8.  het resultaat zal zijn:
9.  --> 4 --> dit is de nieuwe lengte van de array
10. de inhoud van de array na push:
11. Jan, Piet, Joris, Korneel
12. */
```

Reverse

De reverse methode geeft de inhoud van de array terug, in omgekeerde volgorde. (van achter naar voor).

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris"];
3.
4. // geef de inhoud in omgekeerde volgorde
5. document.write(studenten.reverse());
6.
7. /*
8.  het resultaat zal zijn:
9.  --> Joris, Piet, Jan
10. */
```

Shift

Met de shift methode, kan je het eerste element uit de array verwijderen. Deze functie geeft het verwijderde item terug als return waarde.

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris", "Korneel"];
3.
4. // verwijder het eerste element uit de array
5. document.write(studenten.shift());
6.
7. /*
8.  het resultaat zal zijn:
9.  --> Jan
10. de nieuwe inhoud na de shift:
11. Piet, Joris, Korneel
12. */
```

Slice

De slice methode geeft je de mogelijkheid om een deel van de array te kopiëren naar een nieuwe array. Er zijn 2 parameters: de startpositie en de eindpositie (deze is optioneel). Negatieve start en/of eind posities betekenen dat je omgekeerd begint te tellen (van achter naar voor).

Indien je geen eindpositie geeft, gaat hij door tot het einde van de array.

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris", "Korneel"];
3.
4. // kopieer elementen Piet en Joris naar nieuwe array
5. var nieuweArray = studenten.slice(1,3);
6.
7. /*
8.  het resultaat zal een nieuwe array 'nieuweArray' zijn, met volgende inhoud
9.  Piet, Joris
10. */
```

Sort

De sort functie zal je array sorteren. Standaard zal de array gesorteerd worden als tekst, en van a naar z. Je kan de sorteervolgorde, en de manier waarop (bv als getallen) specificeren via een functie (sortfunction)

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris", "Korneel"];
3.
4. // sorteer de array
5. document.write(studenten.sort());
6.
7. /*
8.  het resultaat zal zijn:
9.  Jan, Joris, Korneel, Piet
10. */
```

En een voorbeeld waarbij we getallen willen laten sorteren:

```
1. // maak een array aan
2. var punten = [40,100,1,5,25,10];
3.
4. // sorteer de array
5. document.write(punten.sort(function(a,b){return a-b}));
6.
7. /*
8.  het resultaat zal zijn:
9.  1,5,10,25,40,100
10. */
```

Hier geven we een sorteerfunctie mee met de sort().

In deze sorteerfunctie zie je dat hij 2 getallen tegen elkaar afweegt. Als het resultaat van a-b positief is, zal a voor b moeten komen, als a negatief is, zal b voor a komen. Als het resultaat 0 is, zijn ze gelijk, en heeft het geen belang welke van de 2 eerst komt.

Splice

De splice functie laat je toe op een specifieke plek elementen uit je array te verwijderen en er eventueel nieuwe aan toe te voegen (op die plek). Je kan ook geen elementen verwijderen en er enkel toevoegen.

Er zijn enkele parameters:

- De index (sleutel) waar je elementen wil verwijderen of toevoegen
- Hoeveel elementen je wil verwijderen (0 = niets verwijderen)
- Opsomming van de nieuwe elementen. (met komma's gescheiden)

```
• // maak een array aan
• var studenten = ["Jan", "Piet", "Joris"];
•
• // op positie 2 wil ik een element bijvoegen
• studenten.splice(2,0,"Korneel");
• document.write(studenten);
•
• // het resultaat zal zijn:
• //Jan, Piet, Korneel, Joris
•
• // op positie 1 wil ik een element verwijderen
• studenten.splice(1,1);
• document.write(studenten);
• // het resultaat zal zijn:
• //Jan, Korneel, Joris
```


toString

Deze functie zal de array omzetten naar een tekstweergave en dit als teruggave resultaat geven.

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris"];
3.
4. // output de string
5. document.write(studenten.toString());
6.
7.
8. // het resultaat zal zijn:
9. //Jan, Piet, Joris
```

Unshift

Met de unshift functie kan je elementen aan het begin van je array toevoegen. OPGELET: deze functie werkt niet in browsers die javascript 1.6 niet gebruiken (bv. < IE8)

De functie geeft als resultaat een aangepaste array en als teruggave waarde de nieuwe lengte van de array.

```
1. // maak een array aan
2. var studenten = ["Jan", "Piet", "Joris"];
3.
4. // voeg toe aan begin en output de string
5. studenten.unshift("Korneel", "Omer");
6. document.write(studenten);
7.
8. // het resultaat zal zijn:
9. //Korneel, Omer, Jan, Piet, Joris
```

Voorbeeld : Adventure Game



In dit voorbeeld vertrekken we van een 'map' waarin een mannetje kan rondwandelen. Om te kijken, waar het mannetje mag lopen en waar niet, houden we per beeld een map bij, met daarin de elementen die je op de map ziet. Zo zie je bijvoorbeeld een grasplein (daarover kan je lopen), maar bijvoorbeeld ook een muur of een boom (daarover kan je niet lopen).

De volledige code:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title></title>
5.     <style>
6.       body{
7.         margin:0px;
8.         padding: 0px;
9.       }
10.    #container{
11.      margin: 0 auto;
12.      width;;
13.    }
14.    #speelVeld{
15.      width: 960px;
16.    }
17.    #speelVeld img{
18.      float:left;
```

```

19.     }
20.     #personage {
21.         background: url('./images/pokemon_mannekesGroot.png') no-repeat -6px -1px;
22.         width: 60px;
23.         height: 64px;
24.         position: absolute;
25.     }
26.     .tegel{
27.         width: 64px;
28.         height: 64px;
29.     }
30. </style>
31. <script>
32. // credit voor images: http://mystikrpg.com/mapeditor/#
33.
34.
35. var personagelinks = 7, personageTop = 7;
36. var veld;
37. var uitvoer = "";
38. var actiefVeld = 10;
39.
40.
41. /* om te bepalen waar de kaart zich bevindt, tellen we horizontaal 1 bij, verticaal 10
42.    0  1  2  3  ...
43.   10 11 12 13 ...
44.   20 21 22 23 ...
45.   ...
46. */
47.
48. function setupSpeelVeld(){
49.
50.     /* 01 = gras
51.        02 = houtenbrug
52.        03 = asfalt
53.        04 = poort
54.        05 = kassei
55.        06 = ijs (sneeuw)
56.        07 = aarde
57.        08 = beton
58.        09 = zand
59.
60.        10 = fontein
61.        11 = boom
62.        12 = steen
63.        13 = lava
64.        14 = water
65.        15 = tralies
66.
67.     */
68.
69.     veld0 = [
70.         [12,12,12,12,15,12,15,12,12,15,12,15,12,12,12],
71.         [12,03,03,03,03,03,03,03,03,03,03,03,03,15],
72.         [12,03,03,03,03,03,03,03,03,03,03,03,03,12],
73.         [12,03,03,03,12,12,12,12,12,12,12,03,03,04],
74.         [15,03,03,03,12,13,13,02,13,13,12,03,03,12],
75.         [12,03,03,03,12,13,13,02,13,13,12,03,03,15],
76.         [12,03,03,03,12,12,12,03,12,12,12,03,03,12],
77.         [15,03,03,03,03,03,03,03,03,03,03,03,03,15],
78.         [12,03,03,03,03,03,03,03,03,03,03,03,03,12],
79.         [12,12,12,15,12,15,12,15,12,15,12,15,12,04,12]
80.     ];
81.
82.     veld1 = [
83.         [12,11,11,11,11,11,11,11,11,11,11,11,11,11],
84.         [15,01,01,01,01,01,01,01,01,01,01,01,01,11],

```

```

85.         [12,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
86.         [04,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
87.         [12,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
88.         [15,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
89.         [12,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
90.         [15,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
91.         [12,01,01,01,01,01,01,02,01,01,01,01,01,01,11],
92.         [12,11,11,11,11,11,11,11,02,11,11,11,11,11,11],
93.     ];
94.
95.
96.     veld10 = [
97.         [11,11,11,11,11,11,11,11,11,12,12,12,12,04,12],
98.         [11,01,01,01,01,01,01,01,01,12,12,15,12,02,12],
99.         [11,11,01,14,07,14,14,01,01,01,10,14,14,02,02],
100.        [11,01,14,14,07,14,14,14,01,01,02,02,02,02,14],
101.        [11,01,14,14,07,14,14,14,01,01,02,14,14,14,14],
102.        [11,01,14,14,07,14,14,11,01,01,02,11,10,01,14],
103.        [11,01,01,01,02,01,11,01,01,01,02,01,01,01,14],
104.        [11,01,02,02,02,02,02,02,02,02,02,01,01,01,11],
105.        [11,01,02,01,01,01,01,10,01,01,01,01,11,11,11],
106.        [11,11,02,11,11,11,11,11,11,11,11,11,11,11,11],
107.    ];
108.
109.     veld11 = [
110.         [11,11,11,11,11,11,11,02,11,11,11,11,11,11,11],
111.         [11,01,01,01,01,01,01,02,01,01,01,01,01,01,11],
112.         [02,02,02,02,02,02,02,02,01,01,01,01,01,01,11],
113.         [15,15,15,15,15,15,15,04,15,15,15,15,15,15,15],
114.         [12,09,09,09,09,09,09,02,09,09,09,09,09,09,12],
115.         [15,09,09,13,13,13,09,02,09,09,09,09,09,09,12],
116.         [12,09,09,13,13,13,09,02,02,02,02,02,02,09,15],
117.         [15,09,09,13,13,13,09,09,09,09,09,09,02,09,15],
118.         [12,09,09,09,09,09,09,09,09,09,09,09,02,09,12],
119.         [12,15,12,15,12,12,15,12,12,15,12,12,04,12,12],
120.    ];
121.
122.
123.     veld20 = [
124.         [11,11,02,11,11,11,11,11,14,12,12,12,12,12,12],
125.         [11,11,02,01,01,01,01,01,14,12,15,15,15,15,12],
126.         [11,11,02,02,02,01,01,01,14,14,14,14,14,14,14],
127.         [11,11,01,01,02,01,01,01,01,01,01,01,01,01,11],
128.         [11,11,01,01,02,02,02,02,02,02,02,02,02,02,02],
129.         [11,11,01,01,02,02,10,10,02,02,02,01,01,01,11],
130.         [11,11,01,01,02,02,10,10,02,02,02,01,01,01,11],
131.         [11,11,01,01,02,02,02,02,02,02,02,02,01,01,11],
132.         [11,11,01,01,01,01,01,01,01,01,01,01,11,11,11],
133.         [11,11,11,11,11,11,11,11,11,11,11,11,11,11,11],
134.    ];
135.
136.
137.     veld21 = [
138.         [12,14,11,11,11,11,11,11,11,11,12,15,04,15,12],
139.         [12,14,01,01,01,01,01,01,01,01,12,10,02,10,12],
140.         [14,14,01,01,01,01,01,01,01,01,12,10,02,10,12],
141.         [11,01,01,01,01,01,01,01,01,01,12,10,02,10,12],
142.         [02,02,02,02,02,02,02,02,02,01,12,12,04,12,12],
143.         [11,01,01,01,02,02,10,10,02,01,01,01,02,01,11],
144.         [11,01,01,01,02,02,10,10,02,01,02,02,02,01,11],
145.         [11,01,01,01,02,02,02,02,02,02,02,01,01,01,11],
146.         [11,01,01,01,01,01,01,01,01,01,01,01,01,01,11],
147.         [11,11,11,11,11,11,11,11,11,11,11,11,11,11,11],
148.    ];
149.
150.

```

```

151.         veld = veld10;
152.         actiefVeld = 10;
153.
154.         tekenVeld();
155.     }
156.
157.
158.     function zetpersonage(delta){
159.         var personage = document.getElementById("personage");
160.         var tempTop = personageTop, templinks=personageLinks;
161.         /*eerst nakijken of de beweging zou mogen, dan pas uitvoeren...*/
162.         /*test of het kan*/
163.         if (delta == "minTop"){tempTop = (personageTop - 1) ;}
164.         if (delta == "plusTop"){tempTop = (personageTop + 1);}
165.         if (delta == "minLinks"){templinks = (personageLinks - 1);}
166.         if (delta == "plusLinks"){templinks = (personageLinks + 1);}
167.         try{
168.             if (veld[tempTop][templinks] < 10){
169.                 /*het kan, dan mag het geregistreerd worden...*/
170.                 personageTop = tempTop;
171.                 personageLinks = templinks;
172.                 personage.style.left = (personageLinks * 64 + "px");
173.                 personage.style.top = (personageTop * 64) + "px";
174.             }
175.         }catch(err){}
176.
177.         /*Nieuwe map laden naar beneden toe!*/
178.         if (tempTop > 9){
179.             actiefVeld += 10;
180.             veld = eval("veld"+actiefVeld);
181.             try{
182.                 tekenVeld();
183.                 personageTop = 0;
184.                 personage.style.left = (personageLinks * 64 + "px");
185.                 personage.style.top = (personageTop * 64) + "px";
186.                 zetpersonage();
187.             }catch(error){
188.                 // deze map bestaat niet?!
189.             }
190.         }
191.
192.         /*Nieuwe map laden naar boven toe!*/
193.         if (tempTop < 0){
194.             actiefVeld -= 10
195.             veld = eval("veld"+actiefVeld);
196.             try{
197.                 tekenVeld();
198.                 personageTop = 9;
199.                 personage.style.left = (personageLinks * 64 + "px");
200.                 personage.style.top = (personageTop * 64) + "px";
201.                 zetpersonage();
202.             }catch(error){
203.                 // deze map bestaat niet?!
204.             }
205.         }
206.
207.         /*Nieuwe map laden naar rechts toe!*/
208.         if (templinks > 14){
209.             actiefVeld += 1
210.             veld = eval("veld"+actiefVeld);
211.             try{
212.                 tekenVeld();
213.                 personageLinks = 0;
214.                 personage.style.left = (personageLinks * 64 + "px");
215.                 personage.style.top = (personageTop * 64) + "px";
216.                 zetpersonage();

```

```

217.         }catch(error){
218.             // deze map bestaat niet?!
219.         }
220.     }
221.
222.     /*Nieuwe map laden naar links toe!*/
223.     if (templinks < 0){
224.         actiefVeld -= 1
225.         veld = eval("veld"+actiefVeld);
226.         try{
227.             tekenVeld();
228.             personageLinks = 14;
229.             personage.style.left = (personageLinks * 64 + "px");
230.             personage.style.top = (personageTop * 64) + "px";
231.             zetpersonage();
232.         }catch(error){
233.             // deze map bestaat niet?!
234.         }
235.     }
236.
237. }
238.
239. function tekenVeld(){
240.     uitvoer = "";
241.     for (var rij=0;rij<10;rij++){
242.         for (var kolom=0;kolom<15;kolom++){
243.             uitvoer += (veld[rij][kolom] == 01) ? "<img class='tegel' src='./images/gras.png'>":"";
244.             uitvoer += (veld[rij][kolom] == 02) ? "<img class='tegel' src='./images/houtenbrug.png'>":"";
245.             uitvoer += (veld[rij][kolom] == 03) ? "<img class='tegel' src='./images/asfalt.png'>":"";
246.             uitvoer += (veld[rij][kolom] == 04) ? "<img class='tegel' src='./images/poort.png'>":"";
247.             uitvoer += (veld[rij][kolom] == 05) ? "<img class='tegel' src='./images/kassei.png'>":"";
248.             uitvoer += (veld[rij][kolom] == 06) ? "<img class='tegel' src='./images/ijs.png'>":"";
249.             uitvoer += (veld[rij][kolom] == 07) ? "<img class='tegel' src='./images/aarde.png'>":"";
250.             uitvoer += (veld[rij][kolom] == 08) ? "<img class='tegel' src='./images/beton.png'>":"";
251.             uitvoer += (veld[rij][kolom] == 09) ? "<img class='tegel' src='./images/zand.png'>":"";
252.             uitvoer += (veld[rij][kolom] == 10) ? "<img class='tegel' src='./images/fontein.png'>":"";
253.             uitvoer += (veld[rij][kolom] == 11) ? "<img class='tegel' src='./images/boom.png'>":"";
254.             uitvoer += (veld[rij][kolom] == 12) ? "<img class='tegel' src='./images/steen.png'>":"";
255.             uitvoer += (veld[rij][kolom] == 13) ? "<img class='tegel' src='./images/lava.png'>":"";
256.             uitvoer += (veld[rij][kolom] == 14) ? "<img class='tegel' src='./images/water.png'>":"";
257.             uitvoer += (veld[rij][kolom] == 15) ? "<img class='tegel' src='./images/tralies.png'>":"";
258.
259.
260.         }
261.     }
262.
263.     document.getElementById("speelVeld").innerHTML = uitvoer;
264.     document.getElementById("speelVeld").innerHTML += "<div id='personage'></div>";
265.     zetpersonage();
266. }
267.
268.
269. document.onkeydown = function(evt) {
270.     evt = evt || window.event;
271.     switch (evt.keyCode) {
272.         case 38:
273.             upArrowPressed();
274.             break;
275.         case 40:
276.             downArrowPressed();
277.             break;
278.         case 37:
279.             leftArrowPressed();
280.             break;
281.         case 39:
282.             rightArrowPressed();

```

```

283.             break;
284.         }
285.     };
286.
287.
288.     function upArrowPressed() {
289.         var personage = document.getElementById("personage");
290.         personage.style.backgroundColor = "-207px -136px";
291.         setTimeout(function() {personage.style.backgroundColor = "-70px -207px";},145);
292.         zetpersonage("minTop");
293.     }
294.     function downArrowPressed() {
295.         var personage = document.getElementById("personage");
296.         personage.style.backgroundColor = "-6px -1px";
297.         setTimeout(function() {personage.style.backgroundColor = "-68px 0";},145);
298.         zetpersonage("plusTop");
299.     }
300.     function leftArrowPressed() {
301.         var personage = document.getElementById("personage");
302.         personage.style.backgroundColor = "-2px -69px";
303.         setTimeout(function() {personage.style.backgroundColor = "-66px -67px";},145);
304.         zetpersonage("minLinks");
305.     }
306.     function rightArrowPressed() {
307.         var personage = document.getElementById("personage");
308.         personage.style.backgroundColor = "-195px -60px";
309.         setTimeout(function() {personage.style.backgroundColor = "-68px -137px";},145);
310.         zetpersonage("plusLinks");
311.     }
312.
313.     </script>
314. </head>
315. <body onload="setupSpeelVeld()">
316.     <div id="container">
317.         <div id="speelVeld"></div>
318.     </div>
319. </body>
320. </html>

```

Woordje uitleg:

Als het spel start, wordt kaart 10 ingeladen en het mannetje op het veld toegevoegd. De positie van het mannetje houden we steeds bij aan de hand van de coördinaten (links → horizontaal, en top → verticaal).

Je mannetje start op positie 7 van links, en 7 van de top.

Wanneer we op de pijl-toetsen van het keyboard drukken, zal dit opgevangen worden, en zal er gekeken worden naar de positie waarnaar je wil stappen. Dus, stel je wil naar boven, dan zal de coördinaten van de top positie verminderd worden met 1.

Vooraleer we deze beweging gaan uitvoeren, kijken we in de array 'veld' of we deze beweging wel kunnen. Dit wil zeggen, we kijken naar de waarde van het element dat op deze coördinaten in de array. Indien deze waarde kleiner is dan 10, mag de beweging doorgaan, anders niet.

Op deze manier, kan het mannetje over het veld lopen.

Indien het mannetje uit het veld wil lopen, zal er een nieuwe map ingeladen worden.

Om een zekere logica te gebruiken, hou ik bij in welk veld het mannetje momenteel staat.

Dit is het actieve veld.

Indien er een nieuw veld ingeladen wordt naar rechts toe, dan weten we dat het actieve veld, het huidige veldnummer + 1 zal zijn.

Indien het nieuwe veld ingeladen wordt naar onder toe, dan weten we dat het actieve veld, het huidige veldnummer + 10 zal zijn.

Op deze manier, kan je telkens het juiste veld inladen.

Opgelet, als het mannetje de kaart uitloopt, moet je dit mannetje natuurlijk in de andere map aan het begin van de kaart zetten.

→ + 1 en ↓ + 10



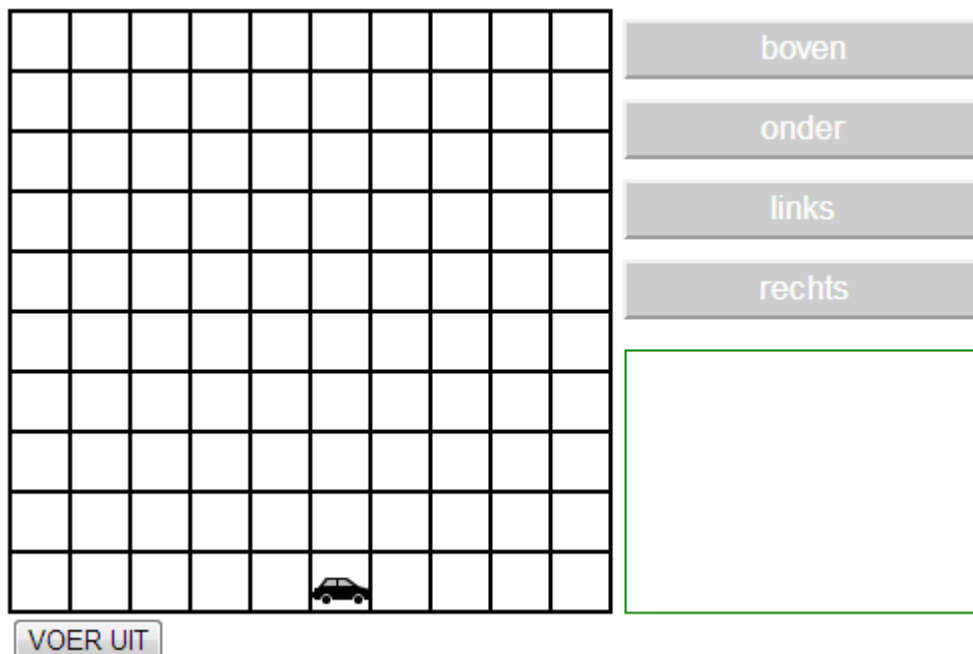
```
veld10 = [  
  [11,11,11,11,11,11,11,11,11,12,12,12,12,04,12],  
  [11,01,01,01,01,01,01,01,01,12,12,15,12,02,12],  
  [11,11,01,14,07,14,14,01,01,01,10,14,14,02,02],  
  [11,01,14,14,07,14,14,14,01,01,02,02,02,02,14],  
  [11,01,14,14,07,14,14,14,01,01,02,14,14,14,14],  
  [11,01,14,14,07,14,14,11,01,01,02,11,10,01,14],  
  [11,01,01,01,02,01,11,01,01,01,02,01,01,01,14],  
  [11,01,02,02,02,02,02,02,02,02,02,01,01,01,11],  
  [11,01,02,01,01,01,01,10,01,01,01,01,11,11,11],  
  [11,11,02,11,11,11,11,11,11,11,11,11,11,11,11]  
];
```


Voorbeeld : Auto

Op het scherm zien we een raster van vakjes, met onderaan een auto.

Hiernaast staan een aantal knoppen (boven, onder, links, rechts en voerUit).

De opdracht was: registreer welke knoppen er geklikt worden en bewaar deze in een array. Wanneer er op de knop 'voer uit' geklikt wordt, moeten de opdrachten die in de array opgeslagen zijn, uitgevoerd worden. Dit wil dus zeggen, dat het autootje, de gevraagde bewegingen achtereenvolgens moet uitvoeren en bewegen over het raster.



De code:

```
1. <!DOCTYPE html>
2. <head>
3. <meta http-equiv="Content-Type" content="text/html; charset=utf8" />
4. <title>auto</title>
5. <style>
6.     #container{
7.         margin: 0 auto;
8.         width: 500px;
9.         height: 300px;
10.        margin: 0 auto;
11.    }
12.    #speelVeld{
13.        width: 300px;
14.        height: 300px;
15.        border: 1px solid black;
```

```

16.     position: relative;
17.     float: left;
18. }
19. .vakje{
20.     width:28px;
21.     height:28px;
22.     border:1px solid black;
23.     float: left;
24. }
25. #auto{
26.     background-image: url("auto.png");
27.     width:28px;
28.     height:28px;
29.     border:1px solid black;
30.     position: absolute;
31.     left:150px;
32.     top:270px;
33. }
34. #commandos{
35.     position: relative;
36.     float: left;
37.     width: 198px;
38.     height: 170px;
39. }
40. .commandoKnop{
41.     width: 180px;
42.     height: 30px;
43.     margin: 5px;
44.     background-color: #ccc;
45.     color: white;
46.     font-size: 13pt;
47. }
48. #commandoLijst{
49.     position: relative;
50.     float: left;
51.     width: 177px;
52.     height: 130px;
53.     border: 1px solid green;
54.     margin-left: 6px;
55. }
56.
57. </style>
58. <script>
59.
60.     var positieTop = 270;
61.     var positieLinks = 150;
62.     var opdrachten = [];
63.     var timer;
64.
65.
66.     function naarBoven(){
67.         if (positieTop > 1){
68.             var auto = document.getElementById("auto");
69.             positieTop -= 30;
70.             opdrachten.push(positieLinks + "|" + positieTop);
71.             //auto.style.top = positieTop + "px";
72.         }
73.     }
74.     function naarOnder(){
75.         if (positieTop < 241){
76.             var auto = document.getElementById("auto");
77.             positieTop += 30;
78.             opdrachten.push(positieLinks + "|" + positieTop);
79.             //auto.style.top = positieTop + "px";
80.         }
81.     }

```

```

82.     function naarLinks(){
83.         if (positieLinks > 1){
84.             var auto = document.getElementById("auto");
85.             positieLinks -= 30;
86.             opdrachten.push(positieLinks + "|" + positieTop);
87.             //auto.style.left = positieLinks + "px";
88.         }
89.     }
90.     function naarRechts(){
91.         if (positieLinks < 241){
92.             var auto = document.getElementById("auto");
93.             positieLinks += 30;
94.             opdrachten.push(positieLinks + "|" + positieTop);
95.             document.getElementById("commandoLijst").innerHTML = opdrachten;
96.             //auto.style.left = positieLinks + "px";
97.         }
98.     }
99.
100.
101.
102.     function start(){
103.         var i = -1;
104.         var int= setInterval(function(){
105.             i++;
106.             var opdracht = opdrachten[i];
107.             var opdrachtSessie = opdracht.split("|");
108.             var links = opdrachtSessie[0];
109.             var top = opdrachtSessie[1];
110.             auto.style.left = links + "px";
111.             auto.style.top = top + "px";
112.
113.             if (i == opdrachten.length-1){
114.                 clearTimeout(int);
115.             }
116.
117.             },300);
118.     }
119.
120. </script>
121. </head>
122.
123. <body>
124.
125.
126.
127.     <div id="container">
128.         <div id="speelVeld">
129.             <div class="vakje"></div><div class="vakje"></div>
130.             <div class="vakje"></div><div class="vakje"></div>
131.             <div class="vakje"></div><div class="vakje"></div>
132.             <div class="vakje"></div><div class="vakje"></div>
133.             <div class="vakje"></div><div class="vakje"></div>
134.             <div class="vakje"></div><div class="vakje"></div>
135.             <div class="vakje"></div><div class="vakje"></div>
136.             <div class="vakje"></div><div class="vakje"></div>
137.             <div class="vakje"></div><div class="vakje"></div>
138.             <div class="vakje"></div><div class="vakje"></div>
139.             <div class="vakje"></div><div class="vakje"></div>
140.             <div class="vakje"></div><div class="vakje"></div>
141.             <div class="vakje"></div><div class="vakje"></div>
142.             <div class="vakje"></div><div class="vakje"></div>
143.             <div class="vakje"></div><div class="vakje"></div>
144.             <div class="vakje"></div><div class="vakje"></div>
145.             <div class="vakje"></div><div class="vakje"></div>
146.             <div class="vakje"></div><div class="vakje"></div>
147.             <div class="vakje"></div><div class="vakje"></div>

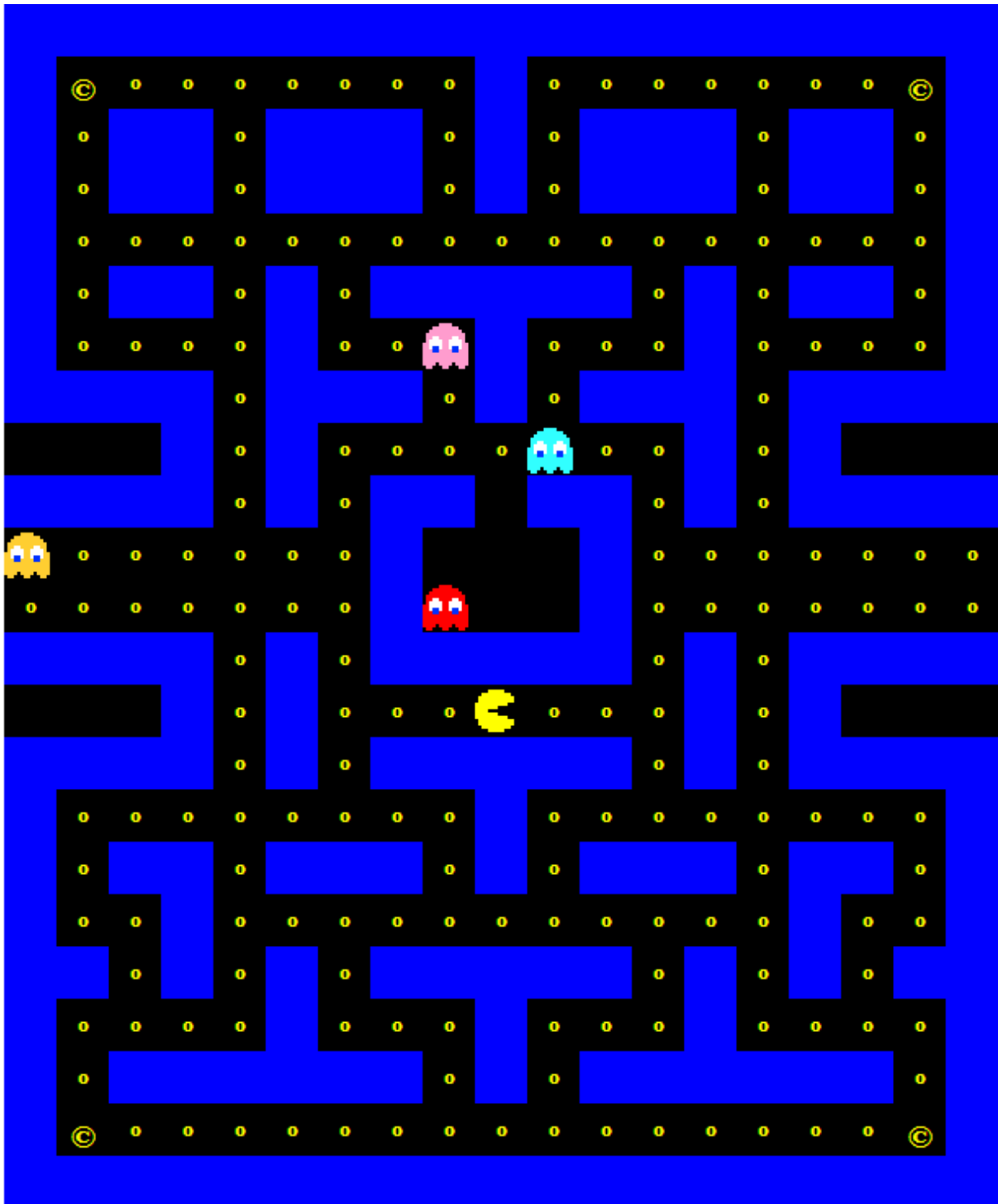
```

```

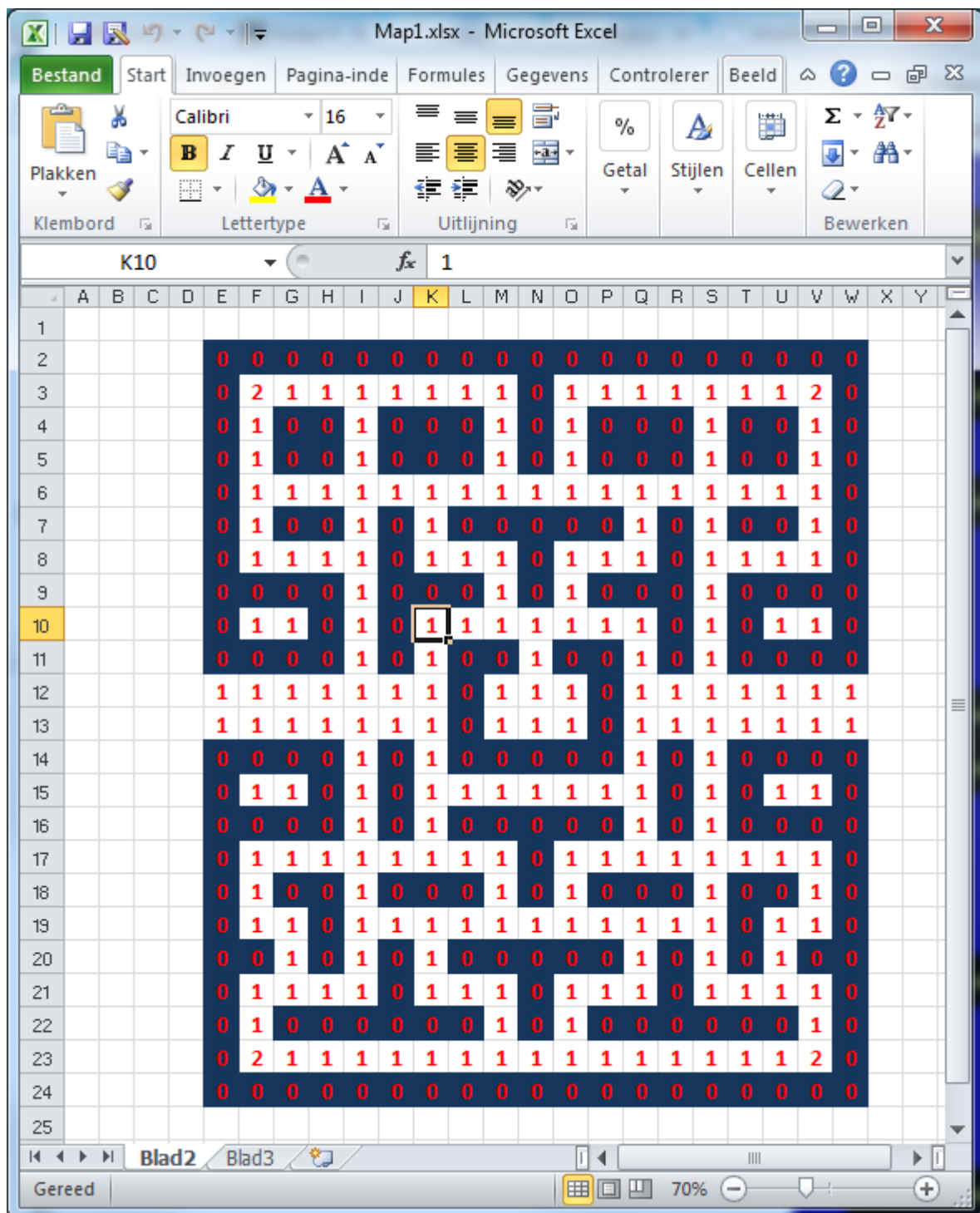
148.         <div class="vakje"></div><div class="vakje"></div>
149.         <div class="vakje"></div><div class="vakje"></div>
150.         <div class="vakje"></div><div class="vakje"></div>
151.         <div class="vakje"></div><div class="vakje"></div>
152.         <div class="vakje"></div><div class="vakje"></div>
153.         <div class="vakje"></div><div class="vakje"></div>
154.         <div class="vakje"></div><div class="vakje"></div>
155.         <div class="vakje"></div><div class="vakje"></div>
156.         <div class="vakje"></div><div class="vakje"></div>
157.         <div class="vakje"></div><div class="vakje"></div>
158.         <div class="vakje"></div><div class="vakje"></div>
159.         <div class="vakje"></div><div class="vakje"></div>
160.         <div class="vakje"></div><div class="vakje"></div>
161.         <div class="vakje"></div><div class="vakje"></div>
162.         <div class="vakje"></div><div class="vakje"></div>
163.         <div class="vakje"></div><div class="vakje"></div>
164.         <div class="vakje"></div><div class="vakje"></div>
165.         <div class="vakje"></div><div class="vakje"></div>
166.         <div class="vakje"></div><div class="vakje"></div>
167.         <div class="vakje"></div><div class="vakje"></div>
168.         <div class="vakje"></div><div class="vakje"></div>
169.         <div class="vakje"></div><div class="vakje"></div>
170.         <div class="vakje"></div><div class="vakje"></div>
171.         <div class="vakje"></div><div class="vakje"></div>
172.         <div class="vakje"></div><div class="vakje"></div>
173.         <div class="vakje"></div><div class="vakje"></div>
174.         <div class="vakje"></div><div class="vakje"></div>
175.         <div class="vakje"></div><div class="vakje"></div>
176.         <div class="vakje"></div><div class="vakje"></div>
177.         <div class="vakje"></div><div class="vakje"></div>
178.         <div class="vakje"></div><div class="vakje"></div>
179.         <div id="auto"></div>
180.     </div>
181.     <div id="commandos">
182.         <button class="commandoKnop" id="naarBoven" onclick="naarBoven()">boven</button>
183.         <button class="commandoKnop" id="naarOnder" onclick="naarOnder()">onder</button>
184.         <button class="commandoKnop" id="naarLinks" onclick="naarLinks()">links</button>
185.         <button class="commandoKnop" id="naarRechts" onclick="naarRechts()">rechts</button>
186.     </div>
187.     <div id="commandoLijst">
188.
189.     </div>
190.     <button onclick="start()">VOER UIT</button>
191. </div>
192.
193. </body>
194. </html>

```

Voorbeeld : Pacman



Het opbouwen van het veld, gebeurde volgens deze logica:



De spel-logica komt eigenlijk altijd terug: er is een timer, die loopt, en bij elk interval van de timer, gebeurt er iets. In dit geval, zal bij elk interval de positie van de spookjes aangepast worden.

Code:

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <title></title>
5.      <style>
6.        body{
7.          margin:0px;
8.          padding: 0px;
9.        }
10.       #container{
11.         margin: 0 auto;
12.         width: 608px;
13.       }
14.       #display{
15.         display : none;
16.         width: 408px;
17.         height: 236px;
18.         font-family: verdana;
19.         font-size: 24pt;
20.         color: red;
21.         background-color: black;
22.         border: 2px solid green;
23.         position: absolute;
24.         margin-left: 100px;
25.         margin-top: 250px;
26.         text-align: center;
27.       }
28.       #display span{
29.         font-size: 14pt;
30.       }
31.       #speelVeld{
32.         width: 608px;
33.         position: absolute;
34.       }
35.       #map{
36.         width: 608px;
37.         height: 736px;
38.         position: absolute;
39.       }
40.       .blokje{
41.         width: 32px;
42.         height: 32px;
43.         float: left;
44.         background-color: black;
45.         color:yellow;
46.         font-size: 15pt;
47.         text-align: center;
48.         line-height: 32pt;
49.         font-weight: bold
50.       }
51.       .muur{
52.         background-color: blue;
53.       }
54.       .spoor{
55.
56.       }
57.       .koek{
58. ;
59.       }
60.       #pacman{
61.         width:32px;
62.         height: 32px;
63.         background: url('./pacmanSprites.png') no-repeat -14px -54px;
```

```

64.         position: absolute;
65.         margin-top: 3px;
66.     }
67.
68.     .spook{
69.         width:32px;
70.         height: 32px;
71.         position: absolute;
72.         margin-top: 3px;
73.     }
74.
75.     #roodSpook{
76.         background: url('./pacmanSprites.png') no-repeat -94px -174px;
77.     }
78.     #blauwSpook{
79.         background: url('./pacmanSprites.png') no-repeat -94px -254px;
80.     }
81.     #geelSpook{
82.         background: url('./pacmanSprites.png') no-repeat -94px -294px;
83.     }
84.     #rozeSpook{
85.         background: url('./pacmanSprites.png') no-repeat -94px -214px;
86.     }
87.
88. </style>
89. <script>
90.
91.     var snelheidSpoken = 200;
92.     var aantalMillisecondenBangeSpoken = 5000;
93.     var spookTimer;
94.     var pauze = false;
95.
96.     var spelBezig = false;
97.     var spokenZijnBang = false;
98.
99.     var pacmanLinks = 9, pacmanTop = 13;
100.    var roodSpookLinks = 9, roodSpookTop = 11;
101.    var geelSpookLinks = 8, geelSpookTop = 11;
102.    var blauwSpookLinks = 10, blauwSpookTop = 11;
103.    var rozeSpookLinks = 9, rozeSpookTop = 10;
104.    var veld;
105.    var uitvoer = "";
106.
107.    var roodSpookDoetMee = true;
108.    var geelSpookDoetMee = true;
109.    var blauwSpookDoetMee = true;
110.    var rozeSpookDoetMee = true;
111.
112.    function setupVeld(){
113.
114.        veld = [
115.            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
116.            [0,2,1,1,1,1,1,1,0,1,1,1,1,1,1,1,2,0],
117.            [0,1,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0],
118.            [0,1,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0],
119.            [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],
120.            [0,1,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0],
121.            [0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,0],
122.            [0,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0],
123.            [3,3,3,0,1,0,1,1,1,1,1,1,0,1,0,3,3,3],
124.            [0,0,0,0,1,0,1,0,3,0,0,1,0,1,0,0,0,0],
125.            [1,1,1,1,1,1,1,0,3,3,3,0,1,1,1,1,1,1],
126.            [1,1,1,1,1,1,1,0,3,3,3,0,1,1,1,1,1,1],
127.            [0,0,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0],
128.            [3,3,3,0,1,0,1,1,3,1,1,1,0,1,0,3,3,3],
129.            [0,0,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0,0],

```



```

130.         [0,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0],
131.         [0,1,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,1,0],
132.         [0,1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,1,0],
133.         [0,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,1,0,0],
134.         [0,1,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,0],
135.         [0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0],
136.         [0,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,0],
137.         [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
138.     ];
139.
140. }
141.
142. function tekenVeld(){
143.     uitvoer = "";
144.     for (var rij=0;rij<23;rij++){
145.         for (var kolom=0;kolom<19;kolom++){
146.             if (veld[rij][kolom] == 0){ // dit is een muur
147.                 uitvoer += "<span class='blokje muur'> </span>";
148.             }
149.             if (veld[rij][kolom] == 1){ // dit is een plaats waar je kan lopen...
150.                 uitvoer += "<span class='blokje spoor'>@</span>";
151.             }
152.             if (veld[rij][kolom] == 2){ // dit is een plaats waar je kan lopen en een grote koek is...
153.                 uitvoer += "<span class='blokje koek'>@</span>";
154.             }
155.             if (veld[rij][kolom] == 3){ // dit is een plaats waar je kan lopen en een grote koek is...
156.                 uitvoer += "<span class='blokje'> </span>";
157.             }
158.         }
159.     }
160.     document.getElementById("map").innerHTML = uitvoer;
161. }
162.
163. function zetPacMan(beweging){
164.
165.     if (!pauze){
166.         //kijken of het spel al begonnen is, anders start je het nu...
167.         if (!spelBezig){
168.             spelBezig = true;
169.             document.getElementById("speelVeld").innerHTML += "<div class='spook' id='roodSpook'></div>";
170.             document.getElementById("speelVeld").innerHTML += "<div class='spook' id='blauwSpook'></div>";
171.             document.getElementById("speelVeld").innerHTML += "<div class='spook' id='geelSpook'></div>";
172.             document.getElementById("speelVeld").innerHTML += "<div class='spook' id='rozeSpook'></div>";
173.             startTimer();
174.         }
175.
176.         pacman = document.getElementById("pacman");
177.         var tempTop = pacmanTop, tempLinks=pacmanLinks;
178.
179.         if (beweging == "minTop"){
180.             tempTop -= 1 ;
181.         }
182.         if (beweging == "plusTop"){
183.             tempTop += 1;
184.         }
185.         if ((beweging == "minLinks") && (pacmanLinks > 0)){
186.             tempLinks -= 1;
187.         }
188.         if ((beweging == "minLinks") && (pacmanLinks == 0)){
189.             tempLinks = 18;
190.         }
191.         if ((beweging == "plusLinks") && (pacmanLinks < 18)){
192.             tempLinks += 1;
193.         }
194.         if ((beweging == "plusLinks") && (pacmanLinks == 18)){
195.             tempLinks = 0;

```

```

196.         }
197.
198.         if (veld[tempTop][tempLinks] != 0){
199.             /*het kan, dan mag het geregistreerd worden...*/
200.             pacmanTop = tempTop;
201.             pacmanLinks = tempLinks;
202.             pacman.style.left = (pacmanLinks * 32 + "px");
203.             pacman.style.top = (pacmanTop * 32) + "px";
204.             if (veld[tempTop][tempLinks] == 1){ // je hebt een kleine koek
205.                 veld[tempTop][tempLinks] = 3;
206.                 tekenVeld();
207.             }
208.             if (veld[tempTop][tempLinks] == 2){ // je hebt een GROTE koek
209.                 veld[tempTop][tempLinks] = 3;
210.                 runSpookrun();
211.                 tekenVeld();
212.             }
213.             kijkOfErGevangenIs();
214.
215.
216.             if ((countElement(1) == 0) && (countElement(2) == 0) && (countElement(4) == 0)){
217.                 // alle velden zijn gedaan... het spel is gedaan...
218.                 document.getElementById("display").innerHTML = "<br><br>WIN WIN WIN <br><span>Druk
op R om opnieuw te spelen</span>";
219.                 document.getElementById("display").style.display = "block";
220.                 clearTimeout(spookTimer);
221.
222.             }
223.
224.
225.
226.         }else{
227.             pacman.style.left = (pacmanLinks * 32 + "px");
228.             pacman.style.top = (pacmanTop * 32) + "px";
229.         }
230.     }
231. }
232.
233. function countElement(item) {
234.     var count = 0;
235.
236.     for (var rij=0;rij<23;rij++){
237.         for (var kolom=0;kolom<19;kolom++){
238.             count += (veld[rij][kolom] == item);
239.         }
240.     }
241.     console.log("aantal elementen: " + item + " ==> " + count);
242.     return count;
243. }
244.
245. function zetSpook(kleur, beweging){
246.
247.     if (kleur == "rood"){
248.         var spook = document.getElementById("roodSpook");
249.         var tempTop = roodSpookTop;
250.         var tempLinks=roodSpookLinks;
251.     }
252.     if (kleur == "geel"){
253.         var spook = document.getElementById("geelSpook");
254.         var tempTop = geelSpookTop;
255.         var tempLinks=geelSpookLinks;
256.     }
257.     if (kleur == "blauw"){
258.         var spook = document.getElementById("blauwSpook");
259.         var tempTop = blauwSpookTop;
260.         var tempLinks=blauwSpookLinks;

```

```

261.         }
262.         if (kleur == "roze"){
263.             var spook = document.getElementById("rozeSpook");
264.             var tempTop = rozeSpookTop;
265.             var tempLinks=rozeSpookLinks;
266.         }
267.
268.
269.
270.
271.         if (beweging == "go"){
272.             var randomNumber = Math.floor((Math.random()*4)+1);
273.
274.         }else{
275.             // gestuurde beweging?
276.         }
277.
278.
279.         if (randomNumber == 1){
280.             tempTop -= 1 ;
281.         }
282.         if (randomNumber == 2){
283.             tempTop += 1;
284.         }
285.         if ((randomNumber == 3) && (tempLinks > 0)){
286.             tempLinks -= 1;
287.         }
288.         if ((randomNumber == 4) && (tempLinks < 18)){
289.             tempLinks += 1;
290.         }
291.
292.         if (veld[tempTop][tempLinks] != 0){
293.             /*het kan, dan mag het geregistreerd worden...*/
294.             if (kleur == "rood"){
295.                 roodSpookTop = tempTop;
296.                 roodSpookLinks = tempLinks;
297.             }
298.             if (kleur == "geel"){
299.                 geelSpookTop = tempTop;
300.                 geelSpookLinks = tempLinks;
301.             }
302.             if (kleur == "blauw"){
303.                 blauwSpookTop = tempTop;
304.                 blauwSpookLinks = tempLinks;
305.             }
306.             if (kleur == "roze"){
307.                 rozeSpookTop = tempTop;
308.                 rozeSpookLinks = tempLinks;
309.             }
310.             spook.style.left = (eval(kleur+"SpookLinks") * 32) + "px";
311.             spook.style.top = (eval(kleur+"SpookTop") * 32) + "px";
312.             kijkOfErGevangenIs();
313.         }else{
314.             /*roodSpook.style.left = (roodSpookLinks * 32 + "px");
315.             roodSpook.style.top = (roodSpookTop * 32) + "px";*/
316.             zetSpook(kleur,beweging);
317.         }
318.
319.     }
320.
321.     function runSpookrun(){
322.         spokenZijnBang = true;
323.         if(roodSpookDoetMee) roodSpook.style.backgroundPosition = "-54px -334px";
324.         if(geelSpookDoetMee)geelSpook.style.backgroundPosition = "-54px -334px";
325.         if(blauwSpookDoetMee)blauwSpook.style.backgroundPosition = "-54px -334px";
326.         if(rozeSpookDoetMee)rozeSpook.style.backgroundPosition = "-54px -334px";

```

```

327.
328.         setTimeout(function() {
329.             if(roodSpookDoetMee)roodSpook.style.backgroundPosition = "-94px -174px";
330.             if(blauwSpookDoetMee)blauwSpook.style.backgroundPosition = "-94px -254px";
331.             if(geelSpookDoetMee)geelSpook.style.backgroundPosition = "-94px -294px";
332.             if(rozeSpookDoetMee)rozeSpook.style.backgroundPosition = "-94px -214px";
333.             spokenZijnBang = false;
334.         },
335.         aantalMillisecondenBangeSpoken
336.     );
337. }
338.
339.
340.
341.     function startTimer(){
342.         //zetSpook("go");
343.         spookTimer = setInterval(
344.             function(){
345.                 if(roodSpookDoetMee)zetSpook("rood", "go");
346.                 if(blauwSpookDoetMee)zetSpook("blauw", "go");
347.                 if(geelSpookDoetMee)zetSpook("geel", "go");
348.                 if(rozeSpookDoetMee)zetSpook("roze", "go")
349.             },
350.             snelheidSpoken
351.         );
352.     }
353.
354.
355.     document.onkeydown = function(evt) {
356.         evt = evt || window.event;
357.         switch (evt.keyCode) {
358.             case 38:
359.                 upArrowPressed();
360.                 break;
361.             case 40:
362.                 downArrowPressed();
363.                 break;
364.             case 37:
365.                 leftArrowPressed();
366.                 break;
367.             case 39:
368.                 rightArrowPressed();
369.                 break;
370.             case 82: // de R is gedrukt
371.                 resetGame();
372.                 break;
373.             case 80: // de P is gedrukt
374.                 pauzeGame();
375.                 break;
376.         }
377.     };
378.
379.
380.     function upArrowPressed() {
381.         var pacman = document.getElementById("pacman");
382.         pacman.style.backgroundPosition = "-54px -102px";
383.         setTimeout(function() {pacman.style.backgroundPosition = "-14px -96px";},145);
384.         zetPacMan("minTop");
385.     }
386.     function downArrowPressed() {
387.         var pacman = document.getElementById("pacman");
388.         pacman.style.backgroundPosition = "-54px -132px";
389.         setTimeout(function() {pacman.style.backgroundPosition = "-14px -134px";},145);
390.         zetPacMan("plusTop");
391.     }
392.     function leftArrowPressed() {

```

```

393.         var pacman = document.getElementById("pacman");
394.         pacman.style.backgroundPosition = "-62px -14px";
395.         setTimeout(function() {pacman.style.backgroundPosition = "-16px -14px";},145);
396.         zetPacMan("minLinks");
397.     }
398.     function rightArrowPressed() {
399.         var pacman = document.getElementById("pacman");
400.         pacman.style.backgroundPosition = "-54px -54px";
401.         setTimeout(function() {pacman.style.backgroundPosition = "-14px -54px";},145);
402.         zetPacMan("plusLinks");
403.     }
404.
405.     function kijkOfErGevangenIs(){
406.         if (((pacmanLinks == roodSpookLinks) && (pacmanTop == roodSpookTop) && roodSpookDoetMee) ||
407.
408.             ((pacmanLinks == geelSpookLinks) && (pacmanTop == geelSpookTop) && geelSpookDoetMee) ||
409.
410.             ((pacmanLinks == blauwSpookLinks) && (pacmanTop == blauwSpookTop) && blauwSpookDoetMee)
411.             ||
412.             ((pacmanLinks == rozeSpookLinks) && (pacmanTop == rozeSpookTop) && rozeSpookDoetMee)
413.         ){
414.             // er is een botsing tussen pacman en een spook !!
415.             if (spokenZijnBang){
416.                 // hoera ! spook gevangen !!
417.                 // extra punten... en het spook verwijderen??
418.                 if ((pacmanLinks == roodSpookLinks) && (pacmanTop == roodSpookTop)){
419.                     roodSpookDoetMee = false;
420.                     document.getElementById("speelVeld").removeChild(roodSpook);
421.                     tekenVeld();
422.                 }
423.                 if ((pacmanLinks == geelSpookLinks) && (pacmanTop == geelSpookTop)){
424.                     geelSpookDoetMee = false;
425.                     document.getElementById("speelVeld").removeChild(geelSpook);
426.                     tekenVeld();
427.                 }
428.                 if ((pacmanLinks == blauwSpookLinks) && (pacmanTop == blauwSpookTop)){
429.                     blauwSpookDoetMee = false;
430.                     document.getElementById("speelVeld").removeChild(blauwSpook);
431.                     tekenVeld();
432.                 }
433.                 if ((pacmanLinks == rozeSpookLinks) && (pacmanTop == rozeSpookTop)){
434.                     rozeSpookDoetMee = false;
435.                     document.getElementById("speelVeld").removeChild(rozeSpook);
436.                     tekenVeld();
437.                 }
438.             }else{
439.                 document.getElementById("display").innerHTML = "<br><br>GAME OVER<br><span>Druk op
440.                 R om opnieuw te spelen</span>";
441.                 document.getElementById("display").style.display = "block";
442.                 clearTimeout(spookTimer);
443.             }
444.         }
445.     }
446.
447.     function resetGame(){
448.         if (pauze) pauzeGame();
449.         clearTimeout(spookTimer);
450.         document.getElementById("display").style.display = "none";
451.         document.getElementById("speelVeld").innerHTML = "";
452.         document.getElementById("map").innerHTML = "";
453.         setupVeld();
454.         spelBezig = false;
455.         pacmanLinks = 9, pacmanTop = 13;
456.         roodSpookLinks = 9, roodSpookTop = 11;
457.         geelSpookLinks = 8, geelSpookTop = 11;
458.         blauwSpookLinks = 10, blauwSpookTop = 11;

```

```

455.         rozeSpookLinks = 9, rozeSpookTop = 10;
456.         roodSpookDoetMee = true;
457.         geelSpookDoetMee = true;
458.         blauwSpookDoetMee = true;
459.         rozeSpookDoetMee = true;
460.         uitvoer = "";
461.         tekenVeld();
462.         document.getElementById("speelVeld").innerHTML += "<div id='pacman'></div>";
463.         zetPacMan();
464.     }
465.
466.     function pauzeGame(){
467.         if (!pauze){
468.             pauze = true;
469.             document.getElementById("display").innerHTML = "<br><br>PAUZE<br><span>Druk op P om ver
der te spelen</span>";
470.             document.getElementById("display").style.display = "block";
471.             clearTimeout(spookTimer);
472.         }else{
473.             pauze = false;
474.             if (spelBezig){
475.                 document.getElementById("display").style.display = "none";
476.                 startTimer();
477.             }
478.         }
479.
480.     }
481.
482.
483.     window.onload = function(){
484.         setupVeld();
485.         tekenVeld();
486.         document.getElementById("speelVeld").innerHTML += "<div id='pacman'></div>";
487.         zetPacMan();
488.     };
489.
490.
491.
492.     </script>
493. </head>
494. <body>
495.     <div id="container">
496.         <div id="map"></div>
497.         <div id="speelVeld"></div>
498.         <div id="display"></div>
499.     </div>
500. </body>
501. </html>

```

Voorbeeld: Puzzel

In dit voorbeeld, gaan we een raster van afbeeldingen tonen, en als er een van deze afbeeldingen aangeklikt wordt, bewaard hij de referentie naar deze afbeelding in een array.

Wanneer er dan een tweede (andere) afbeelding aangeklikt wordt, is het de bedoeling, dat het systeem deze 2 afbeeldingen van plaats wisselt.



Code:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <style>
5.       #container{
6.         width:600px;
7.         height:300px;
```



```

8.         margin:0 auto;
9.         text-align: center;
10.    }
11.    .puzzelStuk{
12.        width: 96px;
13.        height: 96px;
14.        border : 2px solid black;
15.        float: left;
16.    }
17. </style>
18. <script>
19.
20.    var puzzel = [
21.        ["1.png", "2.png", "3.png", "4.png", "5.png", "6.png"],
22.        ["11.png", "12.png", "13.png", "14.png", "15.png", "16.png"],
23.        ["21.png", "22.png", "23.png", "24.png", "25.png", "26.png"]
24.    ];
25.
26.    var selectieLijst = [];
27.
28.
29.    window.onload = function(){
30.        //shuffle(puzzel);
31.        document.getElementById("container").innerHTML = tekenVeld();
32.    }
33.
34.    function tekenVeld(){
35.        uitvoer = "";
36.        for (var rij=0;rij<3;rij++){
37.            for (var kolom=0;kolom<6;kolom++){
38.                uitvoer += "<img onclick='speel(this)' class='puzzelStuk' alt='";
39.                uitvoer += "+rij+"|"+kolom+" src='"+puzzel[rij][kolom]+">";
40.            }
41.            return uitvoer;
42.        }
43.
44.        function speel(puzzelStuk){
45.            // enkel de naam van de file overhouden (dus niet http://localhost.. --
46.            > )
47.            puzzelStukFileName = puzzelStuk.src.substr(puzzelStuk.src.lastIndexOf("
48.            /") + 1);
49.            /*
50.                kijken of er al een element in de selectie zitten,
51.                indien niet --> toevoegen aan de selectie
52.                indien wel --> kijken of het niet hetzelfde element is
53.                indien wel --> niets doen
54.                indien niet --
55.                > dit element wisselen van plek met het element dat in de selectie zit + selectieLi
56.                jst weer leegmaken
57.            */
58.            if (selectieLijst.length > 0){
59.                if (selectieLijst.indexOf(puzzelStukFileName) < 0){
60.                    // het is het 2de element, en het is een ander dan hetgene er a
61.                    l inzit...
62.                    selectieLijst.push(puzzelStukFileName+"|"+puzzelStuk.alt);
63.                    var eersteElement = selectieLijst[0].split("|");
64.                    var tweedeElement = selectieLijst[1].split("|");
65.                    selectieLijst = [];
66.                    puzzel[ tweedeElement[1] ][ tweedeElement[2] ] = eersteElement[0];
67.                    puzzel[ eersteElement[1] ][ eersteElement[2] ] = tweedeElement[0];
68.                    document.getElementById("container").innerHTML = tekenVeld();
69.                }
70.            }else{
71.                selectieLijst.push(puzzelStukFileName+"|"+puzzelStuk.alt);
72.            }
73.        }

```



```
68.     }
69.
70.     </script>
71. </head>
72. <body>
73.
74.     <div id="container">
75.
76.     </div>
77. </body>
78. </html>
```

Timer functies

Timer events in Javascript

- `setInterval()`
- `setTimeout ()`
- `clearTimeout ()`

setInterval

`/* Dit zal elke 3000 milliseconden uitgevoerd worden*/`

```
1. var timer = setInterval(function(){  
2. alert("Hello");  
3. },3000);
```

setTimeout

`/* Dit zal na 3000 milliseconden uitgevoerd worden (1 keer)*/`

```
1. setTimeout(function(){  
2. alert("Hello")  
3. },3000);
```

clearTimeout

`/* Dit stopt de uitvoering van de timer*/`

```
1. clearTimeout(timer);
```

Voorbeeld : de stopwatch



```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <style>
5.       #container{
6.         width:230px;
7.         height:300px;
8.         margin:0 auto;
9.         text-align: center;
10.      }
11.      #klok{
12.        background-image: url('http://www.hometrainingtools.com/images/265/ME-
13.        STOPWAT.jpg');
14.        width: 230px;
15.        height: 265px;
16.      }
17.      #display{
18.        position: relative;
19.        top: 108px;
20.        left: 17px;
21.        font-size: 12pt;
22.        font-family: arial;
23.        font-weight: bold;
24.      }
25.    </style>
26.    <script>
27.      var timer;
28.      var tijd = 0;
29.      var startKnop, resetKnop, display;
30.      var bezig = false;
31.
32.      window.onload = function(){
```

```

33.     startKnop= document.getElementById("startKnop");
34.     resetKnop= document.getElementById("resetKnop");
35.     display = document.getElementById("display");
36.     toonTijd();
37.
38.     startKnop.onclick=function(){
39.         if (! bezig){
40.             timer = window.setInterval(function(){start()},10);
41.             bezig = true;
42.             startKnop.innerHTML = "PAUZE";
43.         }else{
44.             window.clearTimeout(timer);
45.             bezig = false;
46.             startKnop.innerHTML = "START";
47.         }
48.     }
49.     resetKnop.onclick=function(){
50.         tijd = 0;
51.         window.clearTimeout(timer);
52.         bezig = false;
53.         toonTijd();
54.     }
55. }
56.
57. function start(){
58.     tijd++;
59.     toonTijd();
60. }
61.
62. function toonTijd(){
63.
64.     minuten = parseInt((tijd / 6000 ) ); // dit zin de minuten
65.     seconden = parseInt((tijd / 100 )- (minuten * 60)) ; // dit zin de seconden
66.     hondersteVanEenSeconde = parseInt(tijd - (seconden * 100) -
67. (minuten * 6000)) ; // dit zijn de seconden
68.     display.innerHTML = maakDubbel(minuten) + ":" + maakDubbel(seconden) + ":" + maakDubb
69. el(hondersteVanEenSeconde);
70. }
71.
72. function maakDubbel(getal){
73.     if (getal < 10){
74.         return "0" + getal;
75.     }else{
76.         return getal;
77.     }
78. }
79.
80. </script>
81. </head>
82. <body>
83.
84.     <div id="container">
85.         <div id="klok">
86.             <div id="display"></div>
87.             </div>
88.             <button id='startKnop' class="knop">START</button>
89.             <button id='resetKnop' class="knop">RESET</button>
90.         </div>
91.     </body>
92. </html>

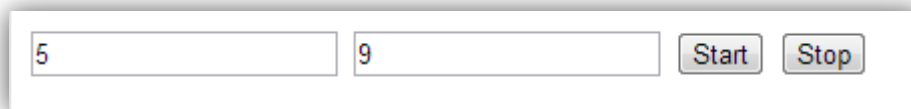
```

Voorbeeld: Keukenwekker

Maak een keukenwekker: 2 velden (Minuten en seconden) en 2 knopjes: start en stop.

Wanneer er op start geklikt wordt, moet de tijd beginnen aftellen. Opgelet: wanneer de seconden op 0 komen, moet er -1 bij de minuten berekend worden...

Wanneer de tijd op 0 komt, moet de timer stoppen. Wanneer er op stop geklikt wordt, moet de timer ook stoppen (pauze). Deze kan je terug hervatten door terug start te klikken)



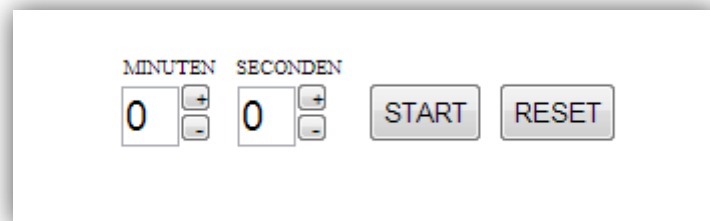
```
1. <html>
2.   <head>
3.     <script>
4.
5.       var keukenwekker;
6.       var isBezig = false;
7.
8.       function startAftel(){
9.
10.        if (isBezig == false){
11.
12.          isBezig = true;
13.          keukenwekker = setInterval(function(){
14.
15.            var minutenInSeconden = parseInt(document.getElementById('Minuten').value) * 60;
16.            var seconden = parseInt(document.getElementById('Seconden').value);
17.            var totaalTijd = minutenInSeconden + seconden;
18.
19.            if (totaalTijd > 0){
20.              totaalTijd--;
21.              var min = parseInt(totaalTijd / 60);
22.              var sec = totaalTijd - (min * 60);
23.              document.getElementById('Minuten').value = min;
24.              document.getElementById('Seconden').value = sec;
25.            }else{
26.              // stop de teller.. we zijn op 0
27.              stopAftel();
28.            }
29.          },1000); // 1000 = 1 sec
30.        }
31.      }
32.
33.      function stopAftel(){
34.        clearTimeout(keukenwekker);
35.        isBezig = false;
36.      }
37.
38.
39.    </script>
```

```

40.     </head>
41.
42.     <body>
43.         <input type='text' id='Minuten' value='0'>
44.         <input type='text' id='Seconden' value='0'>
45.         <button id='startKnop' onclick='startAftel()'>Start</button>
46.         <button id='startKnop' onclick='stopAftel()'>Stop</button>
47.     </body>
48. </html>

```

Eventuele alternatieve oplossing:



MINUTEN SECONDEN

0 0

START RESET

Objecten

Propertes en Methoden

Zoals we al eerder aanhaalden, is in javascript eigenlijk bijna alles een object.

We hebben al gewerkt met Strings (tekst), met getallen, met arrays ...

Al deze zaken, zijn eigenlijk objecten.

Een object bezit meestal een aantal 'properties' of eigenschappen en mogelijk ook een aantal 'methodes' of functies.

Zoals we al eerder gebruikten; bepaal de lengte van een tekst=

```
1. Var tekst = "hallo allemaal";  
2. alert("deze tekst telt zoveel tekens: " + tekst.length );
```

Zoals je kan zien, vragen we aan de hand van de variabele PUNT eigenschap de waarde hiervan op. tekst.length geeft de lengte terug van de tekst.

Een voorbeeld van een Methode is bv. de 'String' methode "toUpperCase()". Deze methode zal de gegeven tekst omzetten naar hoofdletters.

```
1. Var tekst = "hallo allemaal";  
2. alert(tekst.toUpperCase());  
3. // uitvoer == HALLO ALLEMAAL
```

Andere objecten : Math

Het Math object is speciaal gemaakt, voor het werken met mathematische taken.

We hebben dit object al gebruikt, bij oa. Het genereren van een random nummer

Enkele veelgebruikte functies van het Math Object:

Math Object Properties

Property	Description
<u>E</u>	Returns Euler's number (approx. 2.718)
<u>LN2</u>	Returns the natural logarithm of 2 (approx. 0.693)
<u>LN10</u>	Returns the natural logarithm of 10 (approx. 2.302)
<u>LOG2E</u>	Returns the base-2 logarithm of E (approx. 1.442)
<u>LOG10E</u>	Returns the base-10 logarithm of E (approx. 0.434)
<u>PI</u>	Returns PI (approx. 3.14)
<u>SQRT1_2</u>	Returns the square root of 1/2 (approx. 0.707)
<u>SQRT2</u>	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
<u>abs(x)</u>	Returns the absolute value of x
<u>acos(x)</u>	Returns the arccosine of x, in radians
<u>asin(x)</u>	Returns the arcsine of x, in radians
<u>atan(x)</u>	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
<u>atan2(y,x)</u>	Returns the arctangent of the quotient of its arguments
<u>ceil(x)</u>	Returns x, rounded upwards to the nearest integer
<u>cos(x)</u>	Returns the cosine of x (x is in radians)
<u>exp(x)</u>	Returns the value of E ^x
<u>floor(x)</u>	Returns x, rounded downwards to the nearest integer
<u>log(x)</u>	Returns the natural logarithm (base E) of x

<u>max(x,y,z,...,n)</u>	Returns the number with the highest value
<u>min(x,y,z,...,n)</u>	Returns the number with the lowest value
<u>pow(x,y)</u>	Returns the value of x to the power of y
<u>random()</u>	Returns a random number between 0 and 1
<u>round(x)</u>	Rounds x to the nearest integer
<u>sin(x)</u>	Returns the sine of x (x is in radians)
<u>sqrt(x)</u>	Returns the square root of x
<u>tan(x)</u>	Returns the tangent of an angle

Andere objecten : Date

Het Date object is speciaal gemaakt, voor het werken met datum taken.

Enkele veelgebruikte functies van het Date Object:

Method	Description
<u>getDate()</u>	Returns the day of the month (from 1-31)
<u>getDay()</u>	Returns the day of the week (from 0-6)
<u>getFullYear()</u>	Returns the year (four digits)
<u>getHours()</u>	Returns the hour (from 0-23)
<u>getMilliseconds()</u>	Returns the milliseconds (from 0-999)
<u>getMinutes()</u>	Returns the minutes (from 0-59)
<u>getMonth()</u>	Returns the month (from 0-11)
<u>getSeconds()</u>	Returns the seconds (from 0-59)
<u>getTime()</u>	Returns the number of milliseconds since midnight Jan 1, 1970
<u>getTimezoneOffset()</u>	Returns the time difference between UTC time and local time, in minutes
<u>getUTCDate()</u>	Returns the day of the month, according to universal time (from 1-31)

<u>getUTCDay()</u>	Returns the day of the week, according to universal time (from 0-6)
<u>getUTCFullYear()</u>	Returns the year, according to universal time (four digits)
<u>getUTCHours()</u>	Returns the hour, according to universal time (from 0-23)
<u>getUTCMilliseconds()</u>	Returns the milliseconds, according to universal time (from 0-999)
<u>getUTCMinutes()</u>	Returns the minutes, according to universal time (from 0-59)
<u>getUTCMonth()</u>	Returns the month, according to universal time (from 0-11)
<u>getUTCSeconds()</u>	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the <u>getFullYear()</u> method instead
<u>parse()</u>	Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
<u>setDate()</u>	Sets the day of the month of a date object
<u>setFullYear()</u>	Sets the year (four digits) of a date object
<u>setHours()</u>	Sets the hour of a date object
<u>setMilliseconds()</u>	Sets the milliseconds of a date object
<u>setMinutes()</u>	Set the minutes of a date object
<u>setMonth()</u>	Sets the month of a date object
<u>setSeconds()</u>	Sets the seconds of a date object
<u>setTime()</u>	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
<u>setUTCDate()</u>	Sets the day of the month of a date object, according to universal time
<u>setUTCFullYear()</u>	Sets the year of a date object, according to universal time (four digits)
<u>setUTCHours()</u>	Sets the hour of a date object, according to universal time
<u>setUTCMilliseconds()</u>	Sets the milliseconds of a date object, according to universal time
<u>setUTCMinutes()</u>	Set the minutes of a date object, according to universal time
<u>setUTCMonth()</u>	Sets the month of a date object, according to universal time
<u>setUTCSeconds()</u>	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the <u>setFullYear()</u> method instead
<u>toDate()</u>	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the <u>toUTCString()</u> method instead
<u>toISOString()</u>	Returns the date as a string, using the ISO standard

<u>toJSON()</u>	Returns the date as a string, formatted as a JSON date
<u>toLocaleDateString()</u>	Returns the date portion of a Date object as a string, using locale conventions
<u>toLocaleTimeString()</u>	Returns the time portion of a Date object as a string, using locale conventions
<u>toLocaleString()</u>	Converts a Date object to a string, using locale conventions
<u>toString()</u>	Converts a Date object to a string
<u>getTimeString()</u>	Converts the time portion of a Date object to a string
<u>toUTCString()</u>	Converts a Date object to a string, according to universal time
<u>UTC()</u>	Returns the number of milliseconds in a date string since midnight of January 1, 1970, according to universal time
<u>valueOf()</u>	Returns the primitive value of a Date object

Eigen Objecten maken

Maar nog veel interessanter wordt het als we ons eigen object kunnen maken en gebruiken!

Stel je wil informatie over honden wil gebruiken in je script. Dan is het mogelijk interessant om een object “hond” aan te maken. Dit object kan je dan verschillende properties geven, waarin je je data over dit object kan bewaren.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <meta charset="UTF-8">
5.
6.     <script>
7.
8.       var hond = {
9.         aantalLuizen : 4,
10.        kleurPels : "bruin",
11.        kleurOgen : "zwart",
12.        naam : "Blacky",
13.        favorieteEten : "Biefstuk"
14.      };
15.
16.      hond.favorieteEten = "Pizza";
17.
18.      function beschrijfHond(){
19.        return hond.naam + " eet graag " + hond.favorieteEten;
20.      }
21.
22.      function toon(){
23.        document.getElementById("uitvoer").innerHTML = beschrijfHond();
24.      }
25.
26.    </script>
27.
28.  </head>
29.  <body>
30.    <button onclick="toon()">toon mij mijn hond</button>
31.    <div id="uitvoer"></div>
32.  </body>
33. </html>
```

In bovenstaande code kan je zien hoe we een eigen object ‘hond’ aanmaken. Bij de creatie geven we dit object al verschillende properties en hun waarde.

De verschillende properties scheiden we van elkaar doormiddel van een komma. De toewijzing van de waarde aan de property gebeurt aan de hand van een dubbelepunt (:)

Ik maar hier een object hond, met 4 luizen, een bruine pels...

Properties gebruiken

Wanneer je een dergelijk object aangemaakt hebt, kan je aan de hand van de objectnaam.propertynaam de waarde hiervan terug gaan gebruiken, of zetten.

```
1. // geef de naam van de hond
2.
3.   alert(hond.naam);
4.
5.
6. // zet de naam van de hond
7.
8.   hond.naam = "Oscar";
```

Wanneer we dan ook nog eigen methodes willen toevoegen aan ons object, maken we van ons object eigenlijk een functie. Met eigen variabelen (de properties) en eigen functies (de methodes).

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <meta charset="UTF-8">
5.     <script>
6.
7.       var hond = function(n, g, f, a, b){
8.         this.naam = n;
9.         this.geslacht = g;
10.        this.favorieteEten = f;
11.        this.aantalLuizen = a;
12.        this.blafGeluid = b;
13.
14.        this.blaf = function(){
15.          return this.blafGeluid;
16.        }
17.        this.kwispel = function(){
18.          return "swoeshj!!!";
19.        }
20.
21.        this.verandernaam = verandernaam;
22.        function verandernaam (nieuweNaam){
23.          this.naam = nieuweNaam;
24.        }
25.      };
26.
27.      var mijnHond = new hond("blacky","mannelijk","Pizza",4,"blaf blaf");
28.      var jouwHond = new hond("charlie","vrouwelijk","worstjes",0, "woef woef");
29.
30.      mijnHond.naam = "Roger";
31.      mijnHond.verandernaam("Kapitein IGLO");
32.
33.
34.
35.      function beschrijfHond(){
36.        return mijnHond.naam + " eet graag " + mijnHond.favorieteEten +
37.        " en jouw hond " + jouwHond.naam + " eet graag " + jouwHond.favorieteEten+
38.        "<br><br> Jouw hond blaft: " + jouwHond.blaf();
39.      }
```

```

40.
41.     function toon(){
42.         document.getElementById("uitvoer").innerHTML = beschrijfHond();
43.     }
44.
45. </script>
46.
47. </head>
48. <body>
49.     <button onclick="toon()">toon mij mijn hond</button>
50.     <div id="uitvoer"></div>
51. </body>
52. </html>

```

Zoals je hier kan zien, hebben we het object hond nu uitgebreid.

```
var hond = function(n, g, f, a, b){ }
```

Op deze manier, zeggen we dat er een functie bestaat, met naam 'hond'. Deze functie verwacht een aantal invoer-parameters. (n,g,f,a,b). Wanneer deze meegegeven worden, worden deze waardes toegekend aan de interne variabelen.

```

1.     this.naam = n;
2.     this.geslacht = g;
3.     this.favorieteEten = f;
4.     this.aantalLuizen = a;
5.     this.blafGeluid = b;

```

Verder hebben we dan nog enkele methodes toegevoegd. Dit zijn onze 'werkwoorden' van het object.

In het geval voor deze hond, kan deze blaffen en kwispelen.

```

1.     this.kwispel = function(){
2.         return "swoeshj!!!";
3.     }
4.
5.     this.verandernaam = verandernaam;
6.     function verandernaam (nieuweNaam){
7.         this.naam = nieuweNaam;
8.     }

```

Al deze code (de variabelen en de methodes) zitten allen binnen de accolades van de omschrijving van dit object.

Het volledige hond- object zou er als volgt kunnen uitzien:

```
1.     var hond = function(n, g, f, a, b){
2.         this.naam = n;
3.         this.geslacht = g;
4.         this.favorieteEten = f;
5.         this.aantalluizen = a;
6.         this.blafGeluid = b;
7.
8.         this.blaf = function(){
9.             return this.blafGeluid;
10.        }
11.        this.kwispel = function(){
12.            return "swoeshj!!!";
13.        }
14.    };

```

Om de waardes te gebruiken van dit object, hadden we al gezien, dat je dit kan doen, door de object-naam.property-naam te gebruiken. Dit kan je op dezelfde manier ook om deze waarde te wijzigen.

Doch, er bestaat nog een andere manier om de waarde te wijzigen, door middel van een functie:

```
1.  this.verandernaam = verandernaam;
2.  function verandernaam (nieuweNaam){
3.      this.naam = nieuweNaam;
4.  }

```

Je maakt een nieuwe functie aan 'verandernaam' met een verwachte invoer parameter (de nieuwe naam). Om deze te kunnen gebruiken, moet je aan het object nog laten weten dat deze nieuwe functie bestaat. Eenmaal aangemaakt kan je deze functie dan gaan gebruiken.

```
1.  mijnHond.naam = "Roger";
2.  mijnHond.verandernaam("Kapitein IGLO");

```

Hierboven zie je de 2 (gelijkwaardige) statements om de naam van de hond te wijzigen. Het eerste werkt rechtstreeks met het zetten van de property van het object. Het tweede werkt met onze nieuwe gecreëerde functie 'verandernaam'.

Voorbeeld : feest

Maak een object 'persoon', waarin je de data van de personen zal bewaren (naam, voornaam, geslacht, komt). Maak een object 'feest' met de data van een feest (locatie, datum, uur). Schrijf een HTML pagina met javascript waarin je een aantal personen uitnodigt. Deze komen of komen niet, en afhankelijk van of ze komen, toon je ze in een bepaalde lijst.

Extra: voorzie knopjes om ze van de ene lijst naar de andere te kunnen verplaatsen.
Voorzie ook de mogelijkheid om extra genodigden via een form toe te voegen.

Uitvoer:

Ledenlijst voor feest in Het Depot op 22/10/2013 om 22u

Aanwezig:	Afwezig:
1 Martijn Cornelissen komt WEL naar het feest 2 Rina Mertens komt WEL naar het feest 3 Karl Vandertropen komt WEL naar het feest 5 Dieter Thomassen komt WEL naar het feest 6 Tina Mertens komt WEL naar het feest 8 Friea Cornelissen komt WEL naar het feest 9 Marijke Martens komt WEL naar het feest	4 Jan Janssens komt NIET naar het feest 7 Roger Devadder komt NIET naar het feest 10 Kurt Devlies komt NIET naar het feest

naam voornaam ik kom

De geselecteerde persoon...

Code:

```
1.  <html>
2.    <head>
3.      <meta charset="utf-8">
4.      <script>
5.
6.
7.      var persoon = function(volgnr, voornaam, naam, komt){
8.        this.volgnr = volgnr;
9.        this.voornaam = voornaam;
10.       this.naam = naam;
11.       this.komt = komt;
12.
13.       this.schrijfUit = function(){
14.         var welOfNiet = "NIET";
15.         if (this.komt)welOfNiet = "WEL";
16.         return "<option value='"+this.volgnr+"'>"+this.volgnr+ " " + this.voornaam + " " +
17.           this.naam + " komt " + welOfNiet + " naar het feest</option>";
18.       };
19.     }
20.
21.
22.     var feest = {
23.       locatie : "Het Depot",
24.       datum : "22/10/2013",
25.       uur: "22u"
26.     }
27.
28.
29.     var genodigden = [
30.       new persoon(1,"Martijn","Cornelissen",true),
31.       new persoon(2,"Rina","Mertens", true),
32.       new persoon(3,"Karl","Vandertropen",true),
33.       new persoon(4,"Jan","Janssens", false),
34.       new persoon(5,"Dieter","Thomassen",true),
35.       new persoon(6,"Tina","Mertens", true),
36.       new persoon(7,"Roger", "Devadder", false),
37.       new persoon(8,"Friea","Cornelissen",true),
38.       new persoon(9,"Marijke","Martens", true),
39.       new persoon(10,"Kurt", "Devlies", false),
40.     ];
41.
```



```

42.
43.
44.
45.     window.onload=function(){
46.         toonFeest();
47.     }
48.
49.     function toonFeest(){
50.         // indien er al waardes zouden staan, maken we deze eerst leeg...
51.         document.getElementById("aanwezig").innerHTML = "";
52.         document.getElementById("Afwezig").innerHTML = "";
53.
54.         document.getElementById("overzicht").innerHTML = "Ledenlijst voor feest in " +
55.             feest.locatie + " op " + feest.datum + " om " + feest.uur;
56.
57.         for (var i = 0; i < genodigden.length; i++){
58.             var gast = new persoon();
59.             gast = genodigden[i];
60.             // of korter : var gast = genodigden[i];
61.             if (gast.komt)document.getElementById("aanwezig").innerHTML += gast.schrijfUit() + "<br>"
62.             ;
63.             if (!gast.komt)document.getElementById("Afwezig").innerHTML += gast.schrijfUit() + "<br>"
64.             ;
65.         }
66.     }
67.
68.     function voegToe(){
69.         var nieuweGenodigde = new persoon();
70.         nieuweGenodigde.volgnr = genodigden.length + 1;
71.         nieuweGenodigde.naam = document.getElementById("naam").value;
72.         nieuweGenodigde.voornaam = document.getElementById("voornaam").value;
73.         nieuweGenodigde.komt = eval(document.getElementById("komt").value);
74.         genodigden.push(nieuweGenodigde);
75.
76.         toonFeest();
77.
78.         // maak de invulvelden weer leeg...
79.         document.getElementById("naam").value = "";
80.         document.getElementById("voornaam").value = "";
81.         document.getElementById("komt").value = "";
82.     }
83.
84.     function veranderNaarNietKomen(){
85.         var gekozenNr = document.getElementById("aanwezig").value;
86.         for (var t = 0; t < genodigden.length; t++){
87.             if (genodigden[t].volgnr == gekozenNr){
88.                 genodigden[t].komt = false;
89.             }
90.         }
91.         toonFeest();
92.     }
93.
94.     function veranderNaarWelKomen(){
95.         var gekozenNr = document.getElementById("Afwezig").value;
96.         for (var t = 0; t < genodigden.length; t++){
97.             if (genodigden[t].volgnr == gekozenNr){
98.                 genodigden[t].komt = true;
99.             }
100.        }
101.        toonFeest();
102.    }
103.
104.    </script>
105. </head>

```

```

106. <body>
107.     <div>
108.         <H1 id="overzicht"></H1>
109.         Aanwezig:
110.         <select size="10" id="aanwezig" style="width:350px; height:150px;border:1px solid black; over
flow:auto;"></select>
111.         Afwezig:
112.         <select size="10" id="Afwezig" style="width:350px; height:150px;border:1px solid black; overf
low:auto;"></select>
113.         <br><br>
114.         <div>
115.             <input type="text" value="" placeholder="naam" id="naam">
116.             <input type="text" value="" placeholder="voornaam" id="voornaam">
117.             <select id="komt"><option value="true">ik kom</option><option value="false">ik kom niet</op
tion></select>
118.             <button onclick="voegToe()">Voeg Bij</button>
119.             <br><br>
120.             De geselecteerde persoon...
121.             <button onclick="veranderNaarNietKomen()">Komt toch niet</button>
122.             <button onclick="veranderNaarWelKomen()">Komt toch Wel</button>
123.
124.         </div>
125.     </div>
126. </body>
127. </html>

```



Jquery



JQuery en andere frameworks

Jquery is een javascript framework. Een framework is een verzameling van componenten die kunnen gebruikt worden om te programmeren. Het hele voordeel van deze frameworks, is de eenvoud van deze frameworks. Zo zal jQuery een heel aantal van de veel voorkomende problemen oplossen (zoals bv. Browserverschillen, hergebruik van code, prototyping), en het eigenlijke programmeren veel vereenvoudigen.

JQuery

JQuery is gestart in 2006, onder de leiding van **John Resig**. Ondertussen zijn er reeds veel verschillende uitgaves geweest, en evolueert deze framework snel verder. Ondersteuning blijft maar groeien, en er bestaat een zeer actieve userbase met zeer veel beschikbare verschillende plugins en extenties op de code.

JQuery wordt momenteel als een van de leidinggevende frameworks gezien, en wordt ondersteund door de grote softwarebedrijven zoals Google en Microsoft.

Versies

Hoewel er van JQuery sinds de start veel verschillende versies gemaakt zijn, is het belangrijk dat je weet dat er wel wat verschillen zitten tussen de verschillende versies. Als het voor u belangrijk is om ondersteuning te bieden aan 'oudere' browsers (we spreken dan over bv. Internet Explorer 6), mag je geen gebruik maken van de nieuwste versies. De ondersteuning van deze browsers is gestopt vanaf versie 2.0!

Het opgeven van deze browsers liet de makers van jQuery toe om hun code veel te vereenvoudigen. Net voor deze oudere browsers te ondersteunen, was er veel extra code nodig, welke in de nieuwe versies van jQuery dus achterwege gelaten wordt.

jQuery pakketten

Het standaard jQuery pakket bestaat uit een geminimaliseerd javascript bestand (1 bestand) waarin de framework staat. Deze code kan je ofwel downloaden en insluiten in je pagina, ofwel een verwijzing naar deze code zetten in je pagina.

Naast de standaard 'core' code van jQuery bestaan er ondertussen al verschillende 'zij-projecten'. Enkele voorbeelden: jQuery Mobile, jQueryUI...

jQuery in gebruik

Om jQuery te gebruiken, hoeft je niet veel te doen... je moet een verwijzing naar het jQuery code bestand in je html pagina zetten. Ofwel:

- Download je je eigen versie van deze code en bewaart deze in een mapje (bv. Js/jquery.js)
- Ofwel sluit je een verwijzing naar de jquery code in je html pagina.

Het voordeel van het insluiten van een kopie van de code, is dat je zeker bent dat de code steeds beschikbaar zal zijn (je hebt je eigen lokale kopie), en dat deze code niet zal wijzigen.

Het voordeel van een referentie te leggen naar de code van een externe website kan zijn: je werkt steeds met de laatste stabiele versie van jQuery, en over het algemeen werken deze CDN (content delivery network) sneller dan je eigen hosting (dit kan bv. Google of jQuery zelf zijn; welke zeer waarschijnlijk reeds aanwezig is in je browser cache, omdat zoveel websites dit reeds in gebruik hebben).

CDN's

Google

- <http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js>

jQuery:

- <http://code.jquery.com/jquery-latest.js>

Scriptsrc.net:

- <http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js>



<http://scriptsrc.net>

jQuery insluiten

Voorbeeld van een pagina waarin jQuery geladen wordt:

```

1.  <!DOCTYPE HTML>
2.  <html>
3.      <head>
4.          <title>jQuery</title>
5.          <meta charset="utf-8">
6.          <!-- Hier leggen we de link naar de google CDN -->
7.          <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js" type="text/javascript">
8.          </script>
9.
10.         <script type="text/javascript">
11.
12.             $( document ).ready(function() {
13.                 // deze functie wordt opgeroepen wanneer de pagina geladen is.
14.             });
15.
16.         </script>
17.
18.     </head>
19.     <body>
20.
21.         <!-- Hier komt de inhoud van je pagina -->
22.
23.     </body>
24. </html>

```

Wanneer we bovenstaande code bekijken, zien we 2 belangrijke nieuwe aspecten:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js" type="text/javascript"></script>
```

Door deze code in de <head> sectie van je pagina te plakken, zorgen we ervoor dat wanneer de pagina geladen wordt, dit script gedownload wordt van de google; en mee in de pagina beschikbaar komt.

```
1. <script type="text/javascript">
2.
3.     $( document ).ready(function() {
4.         // deze functie wordt opgeroepen wanneer de pagina geladen is.
5.     });
6.
7. </script>
```

De ready() functie van jQuery is een van de meest gebruikte functies. Deze functie is in essentie eigenlijk een andere manier om “window.onload = function(){}” te schrijven.

De code tussen de accolades zal uitgevoerd worden, wanneer de pagina geladen is.

Stel, als test, willen we een popup lagen zien, wanneer de pagina geladen is. Dan kunnen we dit als volgt doen:

```
1. <script type="text/javascript">
2.
3.     $( document ).ready(function() {
4.
5.         alert("test");
6.
7.     });
8.
9. </script>
```

jQuery shorthands

jQuery is eigenlijk een verzameling van javascript functies. Om deze functies te gebruiken, moet je normaal altijd de naam van de functie(object) schrijven, gevolgd door de functie. Dus “jquery()”.

Om dit zo handig en kort mogelijk te maken, hebben de makers van jQuery dit verkort naar “\$”.

Het dollar teken is een verkorte weergave van de oproep naar de jQuery functies.

jQuery Selectors

Selecteren op id

Selectors, of de manier om een bepaald element aan te spreken.

Zoals we tot nu toe altijd werkte met `document.getElementById("idnaamvanelement")`, kunnen we dit veel eenvoudiger in jQuery.

`document.getElementById("idnaamvanelement")` wordt in jQuery `$("#idnaamvanelement")`

Een voorbeeld: Stel, je wil de tekst in een div met id aanpassen naar een andere tekst:

In klassieke javascript:

```
1. <!DOCTYPE HTML>
2. <html>
3.   <head>
4.     <title>jQuery</title>
5.     <meta charset="utf-8">
6.     <script type="text/javascript">
7.       window.onload = function(){
8.         var element = document.getElementById("uitvoer");
9.         element.innerHTML = "Dit is de nieuwe inhoud";
10.      }
11.    </script>
12.  </head>
13.  <body>
14.    <div id="uitvoer"></div>
15.  </body>
16. </html>
```

Het jQuery alternatief:

```
1. <!DOCTYPE HTML>
2. <html>
3.   <head>
4.     <title>jQuery</title>
5.     <meta charset="utf-8">
6.     <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
7.     <script type="text/javascript">
8.       $( document ).ready(function() {
9.         $("#uitvoer").html("Dit is de nieuwe inhoud");
10.      });
11.    </script>
12.  </head>
13.  <body>
14.    <div id="uitvoer"></div>
15.  </body>
16. </html>
```

Besluit:

Als we in jQuery willen verwijzen naar een element met een bepaalde ID, werken we met de hash-tag (#), gevolgd door de id naam zelf.

Selecteren op klasse naam

Wanneer we een reeks van elementen, met een zelfde klasse willen aanspreken (selecteren), kunnen we dit door gebruik te maken van de “punt-notatie”.

```
$(".uitvoerElementen").html("Dit is de nieuwe inhoud");
```

Deze bovenstaande code, zal de inhoud van alle elementen met de klasse ‘uitvoerElementen’, veranderen naar “Dit is de nieuwe inhoud”.

Besluit:

Als we in jQuery willen verwijzen naar een element met een bepaalde klasse, werken we met het punt (.), gevolgd door de naam van de klasse zelf.

Selecteren op element type

Wanneer we een reeks van elementen, van hetzelfde type willen aanspreken (selecteren), kunnen we dit door gebruik te maken van de naam van het type.

```
$("h1").html("Dit is de nieuwe inhoud");
```

Deze bovenstaande code, zal de inhoud van alle elementen van het type “h1”, veranderen naar “Dit is de nieuwe inhoud”.

Besluit:

Als we in jQuery willen verwijzen naar een element van een bepaald type, gebruiken we de naam van het type.



jQuery Events

jQuery heeft een heel aantal ingebouwde functies, klaar om te gebruiken. De meeste van deze functies hebben parameters (eventuele invoer voor de functie), en ook een functie die uitgevoerd wordt wanneer er een bepaald event gebeurt.

Eveneens hebben ze bijna allemaal voorzien ze een callback functie. Deze callback functie zal uitgevoerd worden, wanneer de functie klaar is.

Ready functie

Een van de meest gebruikte hebben we nu gezien, de ready functie. Er zijn een paar verschillende manieren om deze functie aan te roepen, deze zijn allemaal evenwaardig.

```
$( document ).ready(function() {  
    // uitvoeren wanneer de pagina geladen is...  
});
```

```
$(function() {  
    // uitvoeren wanneer de pagina geladen is...  
});
```

```
jQuery( document ).ready(function( $ ) {  
    // uitvoeren wanneer de pagina geladen is...  
});
```

Gebeurtenissen met de muis:

Click()

```
$( "#knop" ).click(function() {  
    alert( "Er is op de knop geklikt..." );  
});
```

Opgelet, deze functie kan je ook gebruiken, om een 'klik' te simuleren (maken):

```
$( "#knop" ).click(function() {  
    $( "#tweedeKnop" ).click();  
});
```

Als er op de knop geklikt wordt, zal dit resulteren in een klik op de tweedeKnop.

Andere functies: Dblclick(), Mouseenter(), Mouseleave(), Mousemove(), Hover(), Mouseout(), Mouseup(), toggle()

Een beetje anders: Hover()

```
$( "#uitvoer" ).hover(  
  function() {  
    $( this ).html( "muis over de div" );  
  }, function() {  
    $( this ).html( "muis NIET over de div" );  
  }  
);
```

Zoals je hierboven kan lezen, bevat de hover functie eigenlijk 2 functies:

Eentje voor wanneer de muis over het element gaat, en eentje voor als de muis het gebied weer verlaten heeft. Je zou deze functie ook in 2 delen kunnen maken met de mouseenter en de mouseleave functies.

Gebeurtenissen met het toetsenbord:

Enkele keyboard functies:

- Focusout()
- Keydown()
- Keypress() (zelfde al keydown, maar geen functie toetsen!)
- keyup()

Gebeurtenissen met het venster (browser window):

Enkele keyboard functies:

- error() → er doet zich een probleem voor
- resize() → de grote van het venster wordt aangepast (vergroot/verkleind)
- scroll() → er wordt gescrolled

Gebeurtenissen met het document:

Enkele keyboard functies:

- load()
- ready()
- unload()

Gebeurtenissen met een formulier:

Enkele keyboard functies:

- blur()
- change()
- focus()
- focusin()
- select()
- submit()

Voor alle events (gebeurtenissen) die momenteel in jQuery zitten, verwijst ik je graag naar de site van jQuery zelf: <http://api.jquery.com/category/events/>

jQuery manipulaties

Net zoals we al eerder deden met javascript, kan je via jQuery de DOM (Document Object Model) manipuleren. Alles wat zich in of aan het browser venster behoort, kan je via de DOM benaderen. De DOM is als het ware een boomstructuur van alle elementen in de browser.

Van alle elementen in de DOM kan je alle eigenschappen (properties) aanpassen.

Enkele voorbeelden van de eigenschappen kunnen zijn:

- de opmaak (de lengte, hoogte, positie, kleur, ...)
- de inhoud (de tekst, de waarde, aangevinkt of niet...)
- ...

Al deze eigenschappen zijn beschikbaar en aanpasbaar via javascript, en dus ook via jQuery.

jQuery Effecten

slideUp

een element naar boven laten dicht schuiven

slideDown

een element naar onder laten open schuiven

toggleSlide

een element laten schuiven, open of dicht, al naargelang de start toestand

Animate

Een animatie laten verlopen (een eigenschap laten veranderen BV; de positie) over een bepaalde tijd.

fadeIn

zichtbaar worden

fadeOut

onzichtbaar worden

toggleFade...

zichtbaar of onzichtbaar worden, al naar gelang de begin toestand

HTML



Als je de browser wil vertellen, dat je zal werken volgens de regels van HTML5, moet je deze regel gebruiken:

Nieuwe 'semantische' elementen

Daarom zijn er verschillende nieuwe elementen bijgekomen. Semantische elementen.

Bv. Section, aside, figure en figcaption, header, footer...

The diagram illustrates a complex web page layout with the following structure:

- body** (outermost container)
 - header**
 - nav**
 - section** (left column)
 - header**
 - nav**
 - footer**
 - section** (right column)
 - header**
 - article**
 - header**
 - p**
 - p**
 - footer**
 - article**
 - header**
 - p** (main content area)
 - aside** (side content area)
 - footer**
 - footer**
 - footer** (bottom-most container)

HTML 5 Audio en Video

HTML5 browsers voorzien een ingebouwde video en audio player. Geen extra plugins nodig.

De code werkt telkens volgens hetzelfde “waterval” principe. Je geeft verschillende bronnen, en deze overloopt hij (de browser) totdat hij eentje tegenkomt die hij kan afspelen.

```
1. <video width="320" height="240" controls id="video1">
2.   <source src="./videos/big_buck_bunny.mp4" type="video/mp4">
3.   <source src="./videos/big_buck_bunny.webm" type="video/webm">
4.   <source src="./videos/big_buck_bunny.ogv" type="video/ogg">
5.   <object width="320" height="240" type="application/x-shockwave-
    flash" data="flowplayer-3.2.1.swf">
6.     <param name="movie" value="flowplayer-3.2.1.swf" />
7.     <param name="allowfullscreen" value="true" />
8.     <param name="flashvars" value="config={'clip': {'url': './videos/big_buck_bunny
    .mp4', 'autoPlay':false, 'autoBuffering':true}}" />
9.     Your browser does not support the video tag.
10.  </object>
11. </video>
12.
```



```
1. <audio id="audio1">
2.   <source src="./audio/FamiliarRoads.ogg" type="audio/ogg" />
3.   <source src="./audio/FamiliarRoads.mp3" type="audio/mpeg" />
4.   <!--<source src="./audio/FamiliarRoads.wav" type="audio/wav" />-->
5.   Your browser does not support HTML5 audio.
6. </audio>
7.
```

```

1. function startAudio() {
2.     var audio1 = document.getElementById("audio1");
3.     if (audio1.paused){
4.         audio1.play();
5.     }else{
6.         audio1.pause();
7.     }
8. }
9. function mute() {
10.    var audio1 = document.getElementById("audio1");
11.    if (audio1.muted){
12.        audio1.muted = false;
13.    }else{
14.        audio1.muted = true;
15.    }
16. }
17.
18. function volumeUp() {
19.    var audio1 = document.getElementById("audio1");
20.    ((audio1.volume+0.1).toFixed(1) <= 1)?audio1.volume = (audio1.volume+0.1).toFixed(1):1;
21. }
22.
23. function volumeDown() {
24.    var audio1 = document.getElementById("audio1");
25.    ((audio1.volume-0.1).toFixed(1) >= 0)?audio1.volume = (audio1.volume-0.1).toFixed(1):0;
26. }

```

HTML 5 Canvas

Het HTML5 Canvas element geeft je de mogelijkheid om te tekenen!

Dit opent vele mogelijkheden (denk aan animaties en spelletjes), waarin telkens de canvas hertekend wordt...

```

1. <canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">
2.     Your browser does not support the HTML5 canvas tag.
3. </canvas>

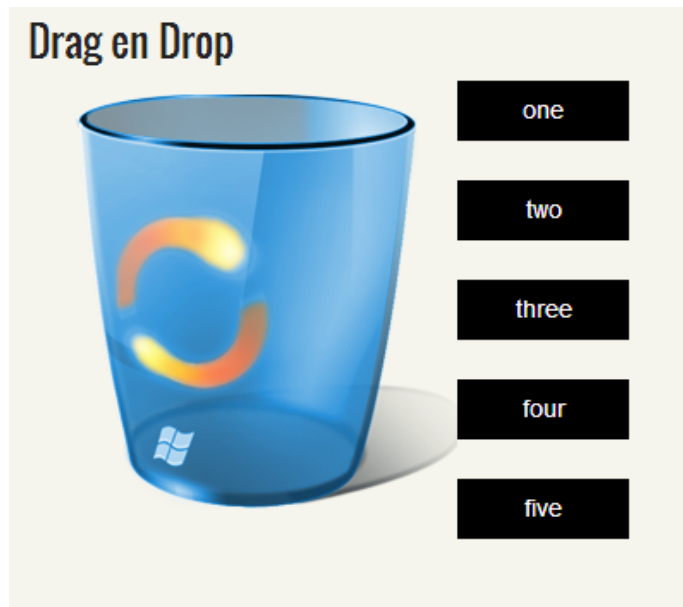
```

```

1. var c=document.getElementById("myCanvas");
2. var ctx=c.getContext("2d");
3. ctx.fillStyle="#FF0000";
4. ctx.fillRect(0,0,150,75);
5. ctx.moveTo(0,0);
6. ctx.lineTo(200,100);
7. ctx.stroke();
8. ctx.beginPath();
9. ctx.arc(95,50,40,0,2*Math.PI);
10. ctx.stroke();
11. ctx.fillStyle="#000000";
12. ctx.font="30px Arial";
13. ctx.fillText("Keep Calm ",10,40);
14. ctx.strokeText("and Carry on",10,80);
15.

```

Drag en Drop



```
1. <div id="vuilbak" ondrop="drop(event)" ondragover="allowDrop(event)" ondragenter="d
   ragenter(event)" ondragleave="dragleave(event)"></div>
2. <div id="slepers">
3.     <span id="one" draggable="true" ondragstart="drag(event)" >one</span><br>
4.     <span id="two" draggable="true" ondragstart="drag(event)" >two</span><br>
5.     <span id="three" draggable="true" ondragstart="drag(event)" >three</span><br>
6.     <span id="four" draggable="true" ondragstart="drag(event)" >four</span><br>
7.     <span id="five" draggable="true" ondragstart="drag(event)" >five</span><br>
8. </div>
```

```
1. function drag(ev){
2.     ev.dataTransfer.effectAllowed = 'copy';
3.     // of move, link, copyLink, copyMove, linkMove, all, none, uninitialized
4.     ev.dataTransfer.setData("Text",ev.target.id);
5. }
6.
7. function allowDrop(ev){
8.     ev.preventDefault();
9. }
10.
11. function drop(ev){
12.     ev.preventDefault();
13.     var data=ev.dataTransfer.getData("Text");
14.     var el = document.getElementById(data);
15.     el.parentNode.removeChild(el);
16.     document.getElementById("vuilbak").style.backgroundColor = '';
17.     document.getElementById("vuilbak").style.backgroundImage = "url('./images/vol.p
   ng')";
18. }
19.
20. function dragenter(e) {
21.     document.getElementById("vuilbak").style.backgroundColor = 'rgba(0,0,0,0.2)';
22.     return false;
23. }
24.
25. function dragleave() {
26.     document.getElementById("vuilbak").style.backgroundColor = '';
27. }
```

GeoLocation

Door gebruik te maken van geolocation, kan je aan de browser vragen “waar ben ik”. Deze zal, aan de hand van de op dat moment beschikbare technologie, een benadering van je huidige positie op aarde geven.

Hoe beter de technologie (als je toestel dit heeft), hoe juister de informatie zal zijn.

Zo kan hij aan de hand van je IP adres, je Wifi station, je GSM signaal, je GPS een positie gaan bepalen.

Opgelet, om dit te kunnen, moet de gebruiker hiervoor wel zijn toestemming geven (popup).

```
1. <script>
2.
3. var options = {
4.   enableHighAccuracy: true,
5.   timeout: 5000,
6.   maximumAge: 0
7. };
8.
9. function success(pos) {
10.   var crd = pos.coords;
11.
12.   uitvoer.innerHTML += 'Je huidige coördinaten:<br>';
13.   uitvoer.innerHTML += 'Latitude : ' + crd.latitude + '<br>';
14.   uitvoer.innerHTML += 'Longitude: ' + crd.longitude + '<br>';
15.   uitvoer.innerHTML += 'Altitude: ' + crd.altitude + '<br>';
16.   uitvoer.innerHTML += 'Nauwkeurig tot op ongeveer ' + crd.accuracy + ' meter.<br>'
17.   ;
18.   uitvoer.innerHTML += 'Nauwkeurigheid naar hoogte tot op ongeveer ' + crd.altitude
19.   Accuracy + ' meter.<br>';
20. };
21.
22. function error(err) {
23.   console.warn('ERROR(' + err.code + '): ' + err.message);
24. };
25.
26. window.onload = function(){
27.   var uitvoer = document.getElementById("uitvoer");
28.   navigator.geolocation.getCurrentPosition(success, error, options);
29. }
30. </script>
```

De positie wordt telkens in coördinaten teruggegeven.

local storage en sessionStorage

in HTML:

```
1. invoer: <input type="text" value="" id="invoer"><button id="voegToeKnop">voegToe</b  
   utton><button id="clearKnop">Reset</button>  
2. <hr>  
3. <div style="width:500px; height:250px; border: 5px solid black;" id="uitvoer"></div  
   >  
4.
```

In JS:

```
1. $(document).ready(function(){  
2.  
3.     if(typeof(Storage)!="undefined"){  
4.         document.getElementById("uitvoerLocaleStorage").innerHTML = localStorage.lijst;  
5.     }  
6.     else{  
7.         document.getElementById("uitvoerLocaleStorage").innerHTML="localstorage werkt h  
   ier niet...";  
8.     }  
9.  
10.    $("#voegToeKnop").click(function(){  
11.        localStorage.lijst += $("#invoerLocalStorage").val() + "<br>";  
12.        document.getElementById("uitvoerLocaleStorage").innerHTML = localStorage.lijst;  
13.    });  
14.  
15.    $("#clearKnop").click(function(){  
16.        localStorage.lijst = '';  
17.        $("#invoerLocalStorage").val("");  
18.        $("#uitvoerLocaleStorage").html(localStorage.lijst);  
19.    });  
20.  
21. });  
22.
```



RESPONSIVE DESIGN

media queries

Om bepaalde css toe te passen, wanneer het scherm binnen bepaalde marge ligt (qua grootte):

De idee is, om per 'breedte' ranges, bepaalde stijlen toe te passen...

in HTML: <meta name="viewport" content="width=device-width, initial-scale=1.0" />

en eventueel ook verwijzen naar een 'reset.css' of de htmlshiv

```
1. @media only screen and (min-width : 1024px){
2.     // deze code zal gebruikt worden, wanneer het scherm groter is dan 1024px breed
3.     // Deze code zou je eventueel ook als standaard code kunnen nemen,
4.     // dan hoeft je uiteraard de media-querie hier niet rond te zetten...
5. }
6.
7. @media only screen and (min-width : 768px) and (max-width : 1024px) {
8.     // deze code zal gebruikt worden, wanneer het scherm groter is dan 768px breed, en kleiner dan 1024px breed
9. }
10.
11. @media only screen and (min-width : 320px) and (max-width : 768px) {
12.     // deze code zal gebruikt worden, wanneer het scherm groter is dan 320px breed, en kleiner dan 768px breed
13. }
14.
15. @media only screen (max-width : 320px) {
16.     // deze code zal gebruikt worden, wanneer het scherm kleiner is dan 320px breed
17. }
```

detectie van een mobiele browser

PHP:Redirecten naar een mobiele pagina, kan je via bv. onderstaand PHP script:

```
1. <?php
2. $useragent=$_SERVER['HTTP_USER_AGENT'];
3. if(preg_match('/(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fennec|hiptop|iemaobile|ip(hone|od|ad)|iris|kindle|lge |maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm( os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6|0)|symbian|treo|up\.(browser|link)|vodafone|wap|windows (ce|phone)|xda|xiino/i',$useragent)||preg_match('/1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a w|abac|ac(er|oo|s\-)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s )|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-|_|a|g|p|s|t)|tp)|hu(aw|tc)|i(\-20|go|ma)|i230|iac( |\-|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\-)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-
```

```

w))|libw|lynx|m1\~w|m3ga|m50\|ma(te|ui|xo)|mc(01|21|ca)|m\~
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\~
| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-2]|n20[2-
3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\~
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|
\~((1-8)|c))|phil|pire|p1(ay|uc)|pn\~2|po(ck|rt|se)|prox|psio|pt\~g|qa\~
a|qc(07|12|21|32|60|\~[2-7])|i\~
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\|sa(ge|ma|mm|ms|ny|va)|sc(01|h\~|oo|p\~
)|sdk\|se(c(\~|0|1)|47|mc|nd|ri)|sgh\~|shar|sie(\~|m)|sk\~
0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\~|v\~
|v )|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\~|tdg\~|tel(i|m)|tim\~|t\~
mo|to(pl|sh)|ts(70|m\~|m3|m5)|tx\~
9|up(\~b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3])\~
v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\~
| )|webc|whit|wi(g |nc|nw)|wmlb|wonu|x700|yas\~|your|zeto|zte\~
/i',substr($useragent,0,4))){
4.         header('Location: http://m.jouwwebsite.be/');
5.     }
6.     ?>

```

JS: Redirecten naar een mobiele pagina, kan je via bv. onderstaand Javascript script:

```

1. <script type="text/javascript">
2.     if (screen.width <= 699) {
3.         document.location = "mobile.html";
4.     }
5. </script>

```