VDO FULLSTACK

JAVASCRIPT BASIS

Colofon

Titel: JAVASCRIPT BASIS

Auteurs: De Nittis Massimo

Uitgever: SyntraPXL

Editor: De Nittis Massimo

Wat is JavaScript?

De basis van elke webpagina is een tekstbestand met HTML code. Deze HTML code verzorgt de structuur en inhoud van de webpagina. CSS wordt gebruikt om webpagina's op te maken. JavaScript is een scripttaal die veel gebruikt wordt om webpagina's interactief te maken en webapplicaties te ontwikkelen.

Client-side

In deze toepassing wordt JavaScript vooral gebruikt in interactieve webpagina's. Net als bij andere scripttalen is er een interpreter nodig om de geprogrammeerde opdrachten uit te voeren. De meeste moderne browsers beschikken over een eigen interpreter voor JavaScript.

Server-side

JavaScript kan ook gebruikt worden voor server-side scripting. De webservers van Netscape waren de eerste die deze ondersteuning boden. Tegenwoordig ondersteunen alle hostingplatform de serverside versie van JS, nl. NodeJS

JavaScript-Toolkits

Een JavaScript-framework is een library (bibliotheek) welke een aantal JavaScript-functies en widgets bevat voor het ontwikkelen van webapplicaties, waarbij vaak de nadruk ligt op AJAX.

Omgeving

Het Internet Mediatype of MIME voor JavaScript-code is application/javascript, hoewel het niet-officiële text/javascript vaker wordt gebruikt. Om JavaScript op te nemen in een webpagina die voldoet aan de standaard voor HTML, moet het type-attribuut expliciet worden opgegeven in de openingstag:

```
<script>
// code
</script>
```

In HTML5 is bepaald dat JavaScript de standaardtaal is en is het lang-attribuut optioneel, net zoals de CDATA-secties:

```
<script>
alert('Hallo, wereld!');
</script>
```

Invoegen van een JavaScript

Het invoegen van JavaScript in een webpagina is niet altijd hetzelfde. In totaal zijn er een viertal manieren om een script in een webpagina in te brengen nml.:

- ➔ in het head gedeelte van de pagina.
- ➔ in het body gedeelte.
- ➔ in het head én het body gedeelte.
- ➔ en in een extern bestand.

Je script staat altijd in een script tag: <script>code</script>

Scripts in het head gedeelte

Het head gedeelte van een pagina wordt het eerst geladen.

Een script dat je in het head gedeelte zet, is opgeladen voordat er code uit het body gedeelte wordt uitgevoerd.
Soms is het vereist om bepaalde JavaScript code in het head gedeelte te zetten, bijvoorbeeld bij het gebruik van functies in JavaScript.
De structuur van een script in het head gedeelte:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>JavaScript Oriëntatie</title>
  <script>
    tekst = "Welkom in de wereld van JavaScript.";
  </script>
</head>
```

Scripts in het body gedeelte

De structuur van een script in het body gedeelte:

```
<body>
  <script type="text/javascript">
    alert(tekst);
  </script>
</body>
</html>
```

Scripts in het head én het body gedeelte

Er is geen limiet aan het aantal scripts dat je in een pagina mag gebruiken.

Je kan dus zowel JavaScript in het head gedeelte en in het body gedeelte van je pagina zetten.

Het komt vaak voor dat er eerst een aantal functies of variabelen in het head gedeelte worden benoemd met een JavaScript-code, waarna ze door een script in het body gedeelte worden opgeroepen en uitgevoerd.

Een extern script maken

Een extern script is JavaScript code die in een apart bestand staat. Dit is een bestand met een bestandsnaam eindigend op .js. Zo'n extern script is vooral handig als je een bepaald script op meerdere pagina's wilt uitvoeren.

Nu kun je gewoon naar het externe bestand verwijzen, zonder dat je de JavaScript code op iedere pagina hoeft in te voeren. In het externe script staat alleen JavaScript code!
Je mag er dus geen HTML tags inzetten.

Dit is een voorbeeld van de inhoud van een .js bestand:

```
// JavaScript Document
document.write("Dit is een extern script!");
```

Het script bevat alleen JavaScript code en geen HTML tags. Om er een JavaScript bestand van te maken, is het nu alleen nog nodig om dit bestand op te slaan als een .js bestand, bijv. extern.js.

Nu kunnen we in de HTML pagina (extern.html) naar het externe script verwijzen door middel van de volgende code:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Extern JavaScript</title>
</head>
<body>
  <p><script type="text/javascript" src="extern.js"></script></p>
</body>
</html>
```

JavaScript uitvoeren en testen

JavaScript wordt gebruikt in webpagina's en worden dus uitgevoerd en getest in een webbrowser. Daar elke browser een eigen JavaScript interpreter heeft, moet je zoals trouwens elke webpagina de test uitvoeren in verschillende browsers.

Chrome inspector

Firefox wordt door de beschikbaarheid van hulpmiddelen voor ontwikkelaars veel gebruikt als testplatform voor webpagina's. Om fouten in chrome op te sporen, gebruik je in het *chrome* menu de opdracht *Webontwikkelaar* > *Webconsole*.

Zo merk je dat in het bestand extern.js op regel 2 een fout optrad.

Weergeven op een HTML pagina

Het volgende voorbeeld toont hoe je aan een HTML pagina (datum.html) een paragraaf met de datum kunt toevoegen:

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>Datum weergeven</title>
</head>
<body>
 <h1>Datum weergeven</h1>
 <script>
          document.write("<p>" + Date() + "</p>");
 </script>
</body>
</html>
```

Dit voorbeeld gebruikt de internationale schrijfwijze om de datum weer te geven. Pas het JavaScript als volgt aan om de lokale datum weer te geven:

```
datum = new Date();
 document.write("<p>" + datum.toLocaleDateString() + "</p>");
```

De eerste regel slaat de datum met de internationale schrijfwijze op.

De opdracht datum.toLocaleDateString() vormt de internationale datum om tot een datum weergegeven volgens de voorschriften van het land en de taal van de browser.

HTML elementen aanpassen

```
<html>
<head>
</head>
<body>
 <h1>Datum weergeven</h1>
 <p id="datum"></p>
 <script>
  datum = new Date();
  document.getElementById("datum").innerHTML = datum.toLocaleDateString();
 </script>
</body>
</html>
```

Om HTML elementen te manipuleren gebruikt JavaScript de DOM methode getElementById(). Deze methode zorgt voor toegang tot het element met het opgegeven id. De eigenschap innerHTML zorgt ervoor dat de inhoud van het element vervangen wordt door de datum.

Let op het feit dat het JavaScript na het <p> element staat. Zo wordt het JavaScript pas uitgevoerd na het aanmaken van het <p> element. M.a.w. het <p> element moet bestaan om het te kunnen aanpassen.

Functies en events (gebeurtenissen)

JavaScript wordt in een HTML pagina uitgevoerd bij het laden van de pagina. Soms willen we JavaScript pas uitvoeren bij een gebeurtenis (bijvoorbeeld bij het klikken op een knop). In zo'n geval plaatsen we de code in een functie.

Events (gebeurtenissen) worden meestal in combinatie met functies gebruikt (een groep opdrachten starten bij een bepaalde gebeurtenis).

Het voorbeeld (event.html) toont het uitvoeren van een functie bij het klikken op een knop:

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>Reactie op gebeurtenis</title>
 <script>
 function  datumWeergeven(){
   datum = new Date();
    document.getElementById("datum").innerHTML = datum.toLocaleDateString();
   }
 </script>
</head>
<body>
 <h1>Datum weergeven</h1>
 <input type="button" onClick="datumWeergeven();" value="Datum weergeven" />
 <p id="datum"></p>
</body>
</html>
```

Hoofdlettergevoelig

De naam van de functie datumWeergeven bevat een hoofdletter. In tegenstelling tot HTML is JavaScript **hoofdlettergevoelig** – m.a.w. let op het gebruik van hoofdletters bij JavaScript opdrachten, variabelen, objecten en functies.

Opdrachten (statements)

JavaScript opdrachten (statements) voeren bepaalde opdrachten uit. De opdracht

```
document.write("Welkom allemaal");
```

plaatst te tekst Welkom allemaal op de webpagina in de browser. Meestal worden uitvoerbare opdrachten afgesloten met een puntkomma. De meeste JavaScript programmeurs vinden dit een goede programmeertechniek. Vandaar dat de meeste voorbeelden op het internet dit ook gebruiken.

Volgens de JavaScript standaard is het gebruik van een puntkomma optioneel en moet de browser het einde van de regel als het einde van de opdracht interpreteren.

Opmerking: met behulp van puntkomma's kan je meerdere opdrachten op één regel plaatsen.

Commentaar

JavaScript commentaar gebruikt je om code leesbaarder te maken.

Commentaarregel

Een enkele commentaarregel start met //.

```
function datumWeergeven(){
  // internationale datum opvragen
  datum = new Date();
  document.getElementById("datum").innerHTML = datum.toLocaleDateString();
```

Meerdere regels commentaar

Het commentaarblok begint met /* en eindigt met */

```
function datumWeergeven(){
 // internationale datum opvragen
 datum = new Date();
 /*
   De inhoud van het element met het id datum krijgt als inhoud de
   datum weergeven naar de normen van het land en de taal.
 */
 document.getElementById("datum").innerHTML = datum.toLocaleDateString();
}
```

Commentaar gebruiken om opdrachten uit te schakelen

Commentaar wordt ook gebruikt om een opdracht voorlopig uit te schakelen (bijvoorbeeld om fouten op te sporen).

```
function datumWeergeven(){
 // internationale datum opvragen
 datum = new Date();
 // alert(datum);
 /*
   De inhoud van het element met het id datum krijgt als inhoud de
   datum weergeven naar de normen van het land en de taal.
 */
 document.getElementById("datum").innerHTML = datum.toLocaleDateString();
}
```

Vanzelfsprekend kan je met /* en */ een volledige blok met opdrachten uitschakelen.

Commentaar op het einde van een regel

Alles wat volgt op de tekens // tot het einde van de regel wordt beschouwd als commentaar.

```
function datumWeergeven(){ // uitvoeren bij het klikken op de knop
  // internationale datum opvragen
  datum = new Date();
  // alert(datum);
  /*
    De inhoud van het element met het id datum krijgt als inhoud de
    datum weergeven naar de normen van het land en de taal.
  */
  document.getElementById("datum").innerHTML = datum.toLocaleDateString();
}
```

Soms is JavaScript in de browser uitgeschakeld

Hoewel het zelden voorkomt, wordt om veiligheidsredenen of om de aantrekkelijkheid van webpagina's fel te verminderen JavaScript in de browser uitgeschakeld. Zo ingestelde browsers voeren de JavaScript niet uit maar tonen de code als tekst op de webpagina.
Om dit te voorkomen, moet je volgens de JavaScript standaard JavaScript voor HTML verbergen met behulp van HTML commentaar.

Plaats juist voor de eerste JavaScript opdracht de HTML commentaar tag <!-- en na de laatste JavaScript opdracht sluit je de HTML commentaar tag met -->.

```
<body>
  <h1>Datum weergeven</h1>
  <p id="datum"></p>
  <script type="text/javascript">
  <!--
    datum = new Date();
    document.getElementById("datum").innerHTML = datum.toLocaleDateString();
  //-->
  </script>
</body>
```

De tekens // voor de --> verhinderen dat JavaScript --> ziet als een opdracht.

Opdrachten

Datum en/of tijd weergeven

Pas de pagina event.html aan zodat er drie knoppen staan. Elke knop start bij een klik erop een eigen functie.

De eerste knop bestaat reeds en start de functie datumWeergeven().

Schrijf een tweede functie om de lokale tijd weer te geven. Deze functie lijkt als twee druppels water op de eerste functie, maar gebruikt in de plaats van datum.toLocaleDateString() om de datum weer te geven, datum.toLocaleTimeString() om de tijd weer te geven.

Maak een knop *Tijd weergeven* waarbij de pas geschreven functie bij het aanklikken wordt uitgevoerd.

Maak een derde functie met bijhorende knop om de datum en de tijd weer te geven. Gebruik in de functie voor het weergeven van de datum en de tijd datum.toLocaleString().

De Kop aanpassen

De Kop 1 (h1) bevat nu *Datum weergeven*.

Verander de inhoud van de h1 tag naar *Datum en/of tijd weergeven*.
Om de kop bij het klikken op een knop aan te passen:

- ➔ Geef je aan de kop tag (h1) het id kop.
- ➔ Voeg je aan de functie datumWeergeven de volgende opdracht toe:

```
document.getElementById("kop").innerHTML = "Datum weergeven";
```

Verander naargelang de knop waarop geklikt wordt, de kop naar *Tijd weergeven* of *Datum en tijd weergeven*.

Javascript afzonderen

Verplaats alle JavaScript code naar een apart bestand en sla dit bestand op als datum.js.
Zorg dat de pagina event.html gebruik maakt van het externe JavaScript bestand datum.js.
Alle oplossingen van opdrachten staan op het internet.

Variabelen

JavaScript variabelen

Variabelen hebben een naam (kort zoals x of beschrijvend zoals voornaam).
De namen van variabelen in JavaScript zijn:

- ➔ hoofdlettergevoelig
- ➔ beginnen met een letter, $ of een underscore.

De waarde van een variabele kan tijdens het uitvoeren van een script veranderen. Daarbij verwijs je naar de naam van de variabele als je de waarde ervan wilt aanpassen of opvragen.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Voornaam</title>
</head>
<body>
  <p><span id="gepensioneerd"></span> werd opgevolgd door <span id="opvolger"></span>.</p>
  <script>
    let voornaam;
    voornaam = "Frank"; document.getElementById("gepensioneerd").innerHTML =
    voornaam; voornaam = "Harry";
    document.getElementById("opvolger").innerHTML = voornaam;
  </script>
</body>
</html>
```

Variabelen declareren

Een variabele aanmaken noemen we declareren. Dit gebeurt met het sleutelwoord **let**.

opgelet: let, const en var hebben een bepaalde “scope” -> to be continued!

```
let x;
let voornaam;
```

Na de declaratie zijn de variabelen leeg (ze hebben nog geen waarde).
Tijdens de declaratie kan je variabelen een waarde geven:

```
let x = 5;
let voornaam = "Frank";
```

Opmerkingen: Tekstwaarden beginnen en eindigen met aanhalingstekens. Indien de tekstwaarde zelf aanhalingstekens bevat, gebruik je enkelvoudige aanhalingstekens.

```
let uitspraak = 'De professor zei: "Ik verzet mij tegen dit maakbaarheidsidealisme."'
```

Een opnieuw gedeclareerde variabele behoudt zijn oorspronkelijke waarde.

Speciale karakters in tekstwaarden (strings)

De backslash (\) wordt gebruikt om speciale karakters die normaal niet toegestaan zijn toch in een tekstwaarde te gebruiken.

```
let tekst="Ze noemen ons de "Vikings" van het noorden.";
```

In JavaScript begint en eindigt een tekstwaarde (string) met enkele of dubbele aanhalingstekens. Dit zorgt ervoor dat in het bovenstaand voorbeeld de variabele tekst de volgende tekstwaarde zal bevatten: Ze noemen ons de.

Om dit te voorkomen plaats je voor de aanhalingstekens in de tekstwaarde een backslash (\). Een karakter na een backslash wordt door JavaScript letterlijk in de tekstwaarde opgenomen en dus niet gezien als het einde van de tekstwaarde:

```
let tekst="Ze noemen ons de \"Vikings\" van het noorden.";
```

De variabel tekst bevat nu: Ze noemen ons de "Vikings" van het noorden.

Lokale variabelen

Een variabele die binnen een functie gedeclareerd wordt, kan enkel in deze functie gebruikt worden. Bij het beëindigen van de functie wordt de lokale variabele vernietigd.

Je kunt dus lokale variabelen met dezelfde naam in verschillende functies gebruiken, zonder dat ze elkaar beïnvloeden, want lokale variabelen worden enkel herkend in de functie waarin ze gedeclareerd werden.

Globale variabelen

Variabelen die buiten een functie gedeclareerd worden zijn globale variabelen. Alle scripts en functies op één webpagina kunnen de globale variabelen gebruiken.
Globale variabelen worden vernietigd bij het verlaten van de webpagina.

Niet gedeclareerde variabelen

Als je een waarde toekent aan een nog niet gedeclareerde variabele dan wordt de variabele automatisch gedeclareerd als globale variabele.

Operatoren

Je kunt bewerkingen (operaties) uitvoeren op en met variabelen.

```
Y = 5;
z = 2;
x = y + z;
```

De = operator wordt gebruikt om waarden aan een variabele toe te kennen.
De + operator wordt gebruikt om waarden samen te tellen.
De waarde van de variabele x is na het uitvoeren van de opdrachten hierboven 7.

Rekenkundige operatoren

Rekenkundige operatoren worden gebruikt om te rekenen met variabelen en/of waarden.

Toewijzingsoperatoren

Toewijzingsoperatoren worden gebruikt om waarden aan variabelen toe te wijzen.

De + operator en strings (tekenreeksen)

De + operator wordt ook gebruikt om string variabelen of tekst waarden aan elkaar te zetten.

```
let begin = "Het is vandaag";
let einde = "mooi weer.";
let zin = begin + einde;
```

Na het uitvoeren van deze opdrachten, bevat de variabele zin de tekst "Het is vandaagmooi weer.". Om een spatie tussen de twee strings te plaatsen, voeg je aan één van de twee strings een spatie toe:

```
let begin = "Het is vandaag ";
let einde = "mooi weer.";
let zin = begin + einde;
```

Of voeg een spatie toe aan de bewerking:

```
let begin = "Het is vandaag";
let einde = "mooi weer.";
let zin = begin + " " + einde;
```

Na het uitvoeren van de aangepaste opdrachten bevat de variabele zin de tekst "Het is vandaag mooi weer.".

Strings en getallen samenvoegen

Bij het samenvoegen van een getal met een string zal het resultaat steeds een string zijn.

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Strings en getallen samenvoegen</title>
</head>
<body>
  <p>Twee getallen optellen: <span id="tweegetallen"></span>.<br />
    Twee tekenreeksen samenvoegen: <span id="tweestrings"></span>.<br />
    Een getal en een tekenreeks samenvoegen: <span id="getalstring"></span>.<br />
    Een tekenreeks en een getal samenvoegen: <span id="stringgetal"></span>.</p>
```

```
  <script> x = 5 + 5;
    document.getElementById("tweegetallen").innerHTML = x;
    let x = "5" + "5";
    document.getElementById("tweestrings").innerHTML = x;
    let x = 5 + "5";
    document.getElementById("getalstring").innerHTML = x;
    let x = "5" + 5;
    document.getElementById("stringgetal").innerHTML = x;
  </script>
</body>
</html>
```

Gebruikersinvoer opvragen

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Formuliervelden</title>
  <script type="text/javascript">
    function bereken() {
      // de invoer in het tekstveld met het id prijs lezen var
      let prijs = document.getElementById("prijs").value;
      // de invoer in het tekstveld met het id aantal lezen
      let aantal = document.getElementById("aantal").value;
      // het resultaat in het tekstveld met het id resultaat plaatsen
      document.getElementById("resultaat").value = prijs * aantal;
    }
  </script>
</head>
<body>
  <p><label for="prijs">Prijs per stuk</label>
    <input type="text" name="prijs" id="prijs" onchange="bereken();" /> x
    <label for="aantal">Aantal</label>
    <input type="text" name="aantal" id="aantal" onchange="bereken();" />
    =
    <input type="text" name="resultaat" id="resultaat" /></p>
```

```
</body>
</html>
```

De opdracht document.getElementById("prijs").value leest welke waarde de gebruikers in het tekstveld met het id prijs heeft ingetypt. Deze waarde kan in een variabele worden opgeslagen om dan verder gebruikt te worden.

Deze opdracht kan je ook gebruiken om waarden in een tekstveld door JavaScript te laten invullen. M.a.w. bekijk deze opdracht als een variabele waarmee je waarden kunt opslaan in en lezen uit tekstvelden.

Waarden die in een tekstveld worden ingetypt zijn altijd strings (tekst). Ook als je alleen getallen intypt. Om met waarden uit een tekstveld te kunnen rekenen, moet je ze naar getallen omzetten. Een voorbeeld:

```
let prijs = parseFloat(document.getElementById("prijs").value);
```

Daarna ben je zeker dat de variabele prijs een getal bevat en geen tekst (string). Als je met gehele getallen zonder komma's werkt, gebruik je beter parseInt(tekst).

Let op het gebruik van onchange bij de tekstvelden. Bij het aanpassen van de waarde van het tekstveld wordt door onchange de functie bereken() uitgevoerd.

De functie bereken wordt uitgevoerd na het intypen van een nieuwe waarde en het verlaten van het tekstveld (m.a.w. als je de cursor buiten het tekstveld plaatst).

Aantal tegels

Om een vloer te betegelen heb je tegels nodig, hoe groter de oppervlakte hoe meer tegels. Vloeren hebben tussen de tegels voegen. Tegels worden verkocht in verpakkingen of palletten met meerdere tegels. Hoe bereken je het aantal benodigde pakketten tegels?

1. Bereken de oppervlakte van de vloer (lengte x breedte).
2. Bereken de lengte en breedte van de tegels met hun voeg (lengte + voegbreedte, breedte + voegbreedte).
3. Bereken de oppervlakte van elke tegel met voeg (lengte x breedte).
4. Bereken het aantal benodigde tegels voor de vloer (oppervlakte van de vloer / oppervlakte van tegel met voeg).

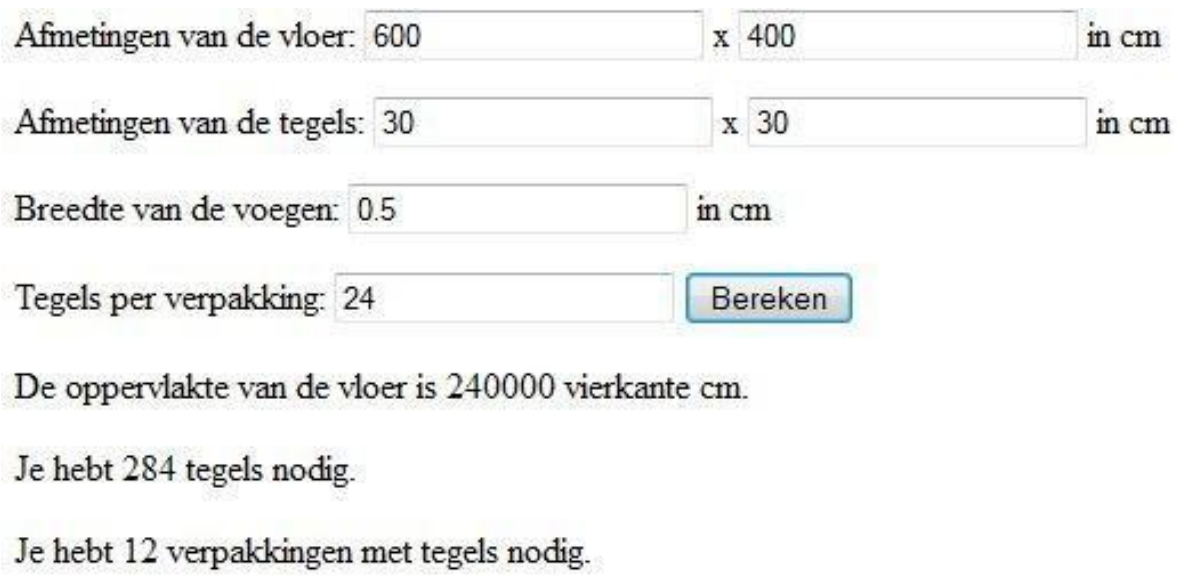
5. Dit is een theoretisch aantal, om verliezen door snijden, e.d.m. in te calculeren neemt men 10 % extra (aantal tegels x 1,1).
6. Bereken het aantal verpakkingen (aantal tegels / aantal tegels per verpakking).

Maak een HTML pagina met tekstvelden om het aantal tegels voor een vloer te berekenen (zie afbeelding).

De resultaten voor het aantal benodigde tegels en verpakkingen zijn meestal kommagetallen. Deze kan je op de volgende manier naar boven afronden:

```
let teBestellenTegels = Math.ceil(tegels);
```

Waarbij de variabele tegels een kommagetal is die naar boven wordt afgerond.



Docent bespreekt de oplossing!

Grafische voorstelling percenten

Maak een HTML pagina met een tekstveld waarin je een percentage kunt ingeven.

Zorg voor een knop waarmee je de aanmaak van de grafische voorstelling met een JavaScript functie kunt starten. Om twee gekleurde balken naast elkaar op het scherm te plaatsen heb je de volgende HTML code nodig:

```
<p>
  <span id="percentage" style="background-color:#ccccff; display:inline-block;
text-align:center"></span>
  <span id="resterend" style="background-color:#ccffcc; display:inline-block;
text-align:center"></span>
</p>
```

Daarbij zorgen CSS stijlen (style) voor de opmaak:

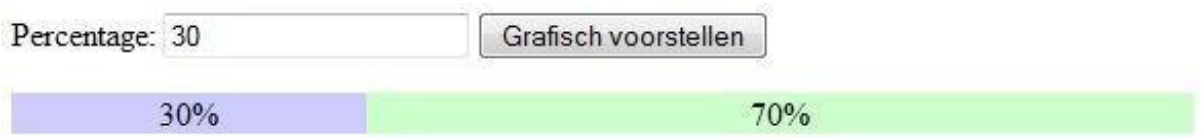
- ➔ background-color: lichtblauwe of lichtgroene achtergrondkleur,
- ➔ display:inline-block: de twee span tags naast elkaar weergeven,
- ➔ text-align:center: de tekst in de span tag centreren.

Schrijf een JavaScript functie waarmee je:

1. het resterende percentage berekent (100 – opgegeven percentage),
2. de breedte van de twee span tags overeenkomt met de percentages:

```
document.getElementById('percentage').style.width = percent + "%";
```

1. de percentages in tekstvorm in de twee tags verschijnt.



Docent bespreekt de oplossing!

JavaScript Controlestructuren

Via controlestructuren kan je de manier waarop je programma wordt uitgevoerd beïnvloeden.

Vergelijkings- en logische operatoren

De vergelijkings- en logische operatoren geven steeds als resultaat juist (true) of fout (false).

Vergelijkingsoperatoren

Vergelijkingsoperatoren onderzoeken de gelijkenissen of verschillen tussen variabelen of waarden. Met **x = 5** toont de tabel de werking van de vergelijkingsoperatoren:

Operator	Beschrijving	Voorbeeld
==	is gelijk aan	x == 8 is fout x == 5 is juist
===	is exact gelijk aan (waarde en type)	x === 5 is juist x === "5" is fout
!=	is niet gelijk aan	x != 8 is juist
>	is groter dan	x > 8 is fout
<	is kleiner dan	x < 8 is juist
>=	is groter dan of gelijk aan	x >= 8 is fout
<=	is kleiner dan of gelijk aan	x <= 8 is juist

Vergelijkingsoperatoren worden gebruikt in voorwaardelijke opdrachten om waarden te vergelijken en op basis van het resultaat een actie te ondernemen:

```
if (leeftijd < 18) alert("Te jong");
```

Logische operatoren

Logische operatoren onderzoeken de logische samenhang van variabelen of waarden.
Met **x = 6** en **y = 3** toont de tabel de werking van logische operatoren:

Operator	Beschrijving	Voorbeeld
&&	en	(x < 10 && y > 1) is juist
||	of	(x == 5 || y == 5) is fout
!	niet	!(x == y) is juist

Voorwaardelijke operatoren

Voorwaardelijke operatoren kennen op basis van een voorwaarde een waarde toe aan een variabele.

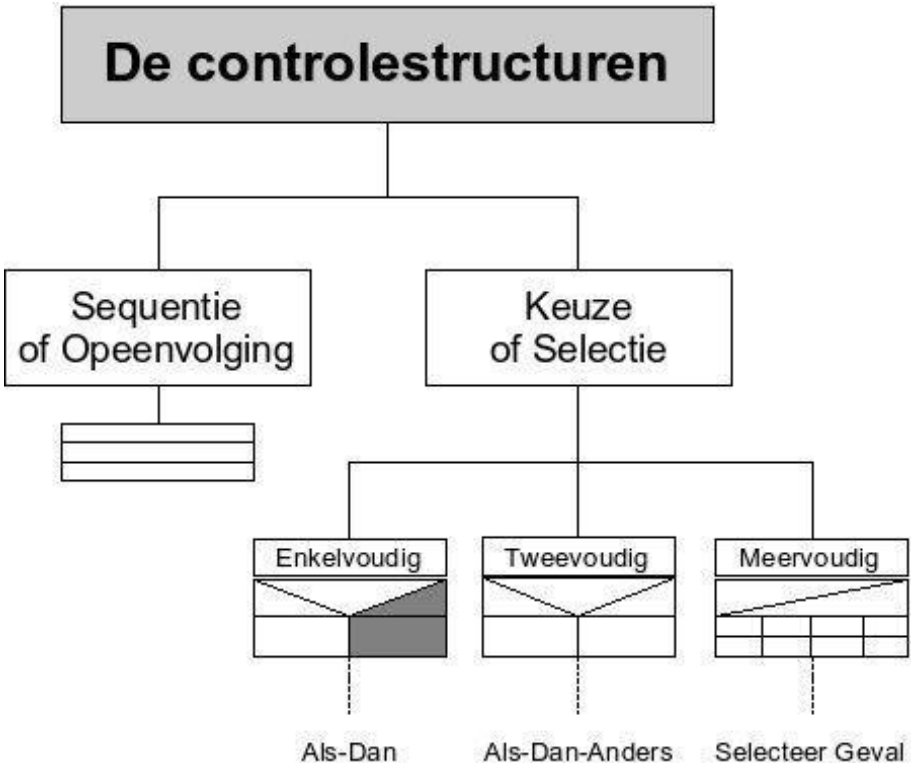
Syntaxis:

variabele=(voorwaarde)?waarde1:waarde2

Voorbeeld:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Welkom</title>
</head>
<body>
  <h1 id="begroeting"></h1>
  <script> var geslacht = "man";
    let naam = "Jansens";
    let begroeting = (geslacht == "man")?"Mijnheer ":"Mevrouw ";
    document.getElementById("begroeting").innerHTML = begroeting + naam;
  </script>
</body>
</html>
```

Voorwaardelijke opdrachten



Voorwaardelijke opdrachten voeren naargelang de voorwaarden verschillende acties uit.
In JavaScript kun je de volgende voorwaardelijke opdrachten gebruiken:

- ➔ **if** opdracht: voert de code enkel uit als aan de voorwaarde is voldaan,
- ➔ **if ... else** opdracht: voert bepaalde code uit als aan de voorwaarde is voldaan en andere code als niet aan de voorwaarde is voldaan,
- ➔ **if ... else if ... else** opdracht: voert één van de vele blokken code uit,
- ➔ **switch** opdracht: voert één van de vele blokken code uit.

If opdracht

Gebruik de if opdracht om de code enkel uit te voeren als aan de voorwaarde is voldaan.

Syntaxis:

```
if (voorwaarde) {
  code die enkel wordt uitgevoerd als aan de voorwaarde is voldaan
}
```

Voorbeeld:

```
<h2 id="goede"></h2>
 <script type="text/javascript">
   // Goedemorgen indien voor 10 uur
   let d = new Date();
   let time = d.getHours();
   if (time < 10){
    document.getElementById("goede").innerHTML = "Goedemorgen.";
   }
 </script>
```

Merk op dat er geen ..else.. in deze code voorkomt. M.a.w. de browser zal de code enkel uitvoeren als aan de conditie is voldaan (voor 10 uur 's morgens).

If ... else opdracht

Gebruik de if ... else opdracht om bepaalde code uit te voeren als aan de voorwaarde is voldaan of andere code uit te voeren als niet aan de voorwaarde is voldaan.

Syntaxis:

```
if (voorwaarde) {
  code wordt uitgevoerd als aan de voorwaarde is voldaan
} else {
  code wordt uitgevoerd als niet aan de voorwaarde wordt voldaan
}
```

Voorbeeld:

```
if (time < 10){
  document.getElementById("goede").innerHTML = "Goedemorgen.";
} else {
  document.getElementById("goede").innerHTML = "Goedendag.";
}
```

If ... else if ... else opdracht

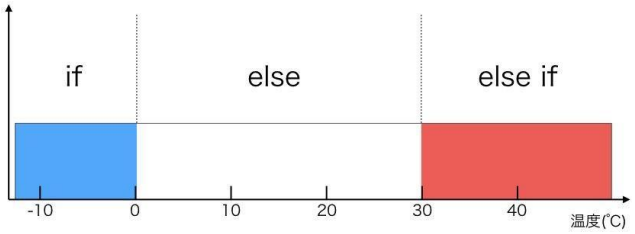
Gebruik de if ... else if ... else opdracht om één van de vele blokken opdrachten uit te voeren.

Syntaxis:

```
if (eerste voorwaarde) {
  code wordt uitgevoerd als aan de eerste voorwaarde is voldaan
} else if (tweede voorwaarde) {
  code wordt uitgevoerd als aan de tweede voorwaarde is voldaan
} else {
  code wordt uitgevoerd als aan geen enkele voorwaarde is voldaan
}
```

Voorbeeld:

```
if (time < 10){
  document.getElementById("goede").innerHTML = "Goedemorgen.";
} else if (time >= 10 && time < 16){
  document.getElementById("goede").innerHTML = "Goedendag.";
} else {
  document.getElementById("goede").innerHTML = "Goedenavond.";
}
```



Switch opdracht

Gebruik de switch opdracht om één van de vele blokken opdrachten uit te voeren.
Syntaxis:

```
switch(n){
  case 1:
    uitvoeren eerste blok opdrachten
    break;
  case 2:
    uitvoeren tweede blok opdrachten
    break;
  default:
    opdrachten die uitgevoerd worden indien n niet gelijk is aan 1 of 2
}
```

Zo werkt het: Eerst hebben we een enkelvoudige bewerking (n) (meestal een variabele) die één keer geëvalueerd wordt. De waarde van de bewerking of variabele wordt daarna vergeleken met elke case waarde in de controlestructuur. Bij een overeenkomst wordt de bijhorende blok opdrachten uitgevoerd. De **break** opdracht sluit de blok uit te voeren opdrachten af. Zonder de break opdracht worden de opdrachten van de volgende case verder uitgevoerd.

Voorbeeld:

```
<p id="weekdag"></p>
<script>
  // begroeting naargelang de weekdag
  let d=new Date();
  let weekdag=d.getDay();
  switch (weekdag){
    case 5:
      document.getElementById("weekdag").innerHTML = "Eindelijk vrijdag.";
      break;
    case 6:
      document.getElementById("weekdag").innerHTML = "Super zaterdag.";
      break;
    case 0:
      document.getElementById("weekdag").innerHTML = "Rustige zondag.";
      break;
```

```
    default:
      document.getElementById("weekdag").innerHTML = "Ik kijk uit naar het weekend.";
  }
</script>
```

Let op:

Als de case waarden teksten (strings) zijn, moet je deze tussen aanhalingstekens plaatsen.

```
case "Nederlands":
```

Opdrachten controlestructuren

Willekeurige koppeling

Met één koppeling willen we op een webpagina twee verschillende sites bereiken.
Op willekeurige basis wordt bij het aanklikken één van beide sites geopend.
Een koppeling (hyperlink) heeft de volgende HTML code:

```
<a id="koppeling" target="_blank">Doe een gok.</a>
```

Om op een willekeurige basis een opdracht te laten uitvoeren, kun je gebruik maken van:

```
let randomNumber = Math.random();
```

Deze opdracht plaatst in de variabele willekeurig een waarde tussen 0 en 1.

Als je deze willekeurige waarde test op groter dan of kleiner dan 0.5, kun je naargelang het resultaat van de test het doel van de koppeling met het id koppeling aanpassen met:

```
document.getElementById('koppeling').href = 'http://www.syntrapxl.be';
```

In het andere geval wordt de koppeling aangepast naar http://www.huppeldepup.be.

Door de keuze van de testwaarde 0.5 heeft elke site 50% kans om als doel voor de koppeling gebruikt te worden.

Drie sites voor één koppeling

Zorg ervoor dat een derde site (gaanmetdiebanaan.be) als doel voor de koppeling gebruikt kan worden. De drie sites moeten evenveel aan bod komen (m.a.w. elk maken ze % kans om als doel gebruikt te worden).

Spreuk van de maand

Elke maand willen we op de webpagina een spreuk van de maand plaatsen.
Maak een HTML pagina met een aanpasbare alinea (p tag).
De JavaScript code bevat:

- ➔ Een variabele datum met als waarde de datum van vandaag.
- ➔ Met de JavaScript code
- ➔ datum.getMonth()
- ➔ kan je de maand uit de variabele datum halen (getallen van 0 tot en met 11, waarbij het getal 0 de maand januari voorstelt).
- ➔ Gebruik deze JavaScript code in een switch opdracht om in de aanpasbare alinea elke maand een passende spreuk te plaatsen.

Functies

Een functie wordt uitgevoerd bij een gebeurtenis (event) of bij een aanroep naar de functie.
Functies worden niet uitgevoerd bij het laden van de webpagina, ze worden enkel gedefinieerd.

Je kunt functies aanroepen vanop elke plaats op de webpagina (zelfs vanaf andere webpagina's als je de functie in een extern .js bestand hebt opgeslagen).
Functies kun je definiëren in de <head> en in de <body> tag van een webpagina (document).
Om er zeker van te zijn dat de functie gedefinieerd is voor hij gebruikt wordt, wordt aangeraden de functie in de <head> van de webpagina te plaatsen.

Functies definiëren

Syntaxis:

```
function functienaam(variabele_1, variabele_2, ..., variabele_X) {
  JavaScript code
}
```

De argumenten (parameters) variabele_1, variabele_2, enz. zijn variabelen of waarden die je aan de functie doorgeeft. De { en de } bepalen het begin en het einde van de functie code.
Opmerkingen:

- ➔ Een functies zonder argumenten heeft achter de functienaam ook de ronde haken () nodig.
- ➔ Let op voor de **hoofdlettergevoeligheid** van JavaScript! Het woord function moet in kleine letters worden geschreven. De naam van de functie mag hoofdletters bevatten, maar moet bij het aanroepen op dezelfde manier inclusief hoofdlettergebruik geschreven worden.

Voorbeeld:

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>Functies</title>
 <script>
          function geklikt(){
            alert("Je hebt op de knop geklikt.");
           }
 </script>
</head>
<body>
 <input type="button" value="Klik hier" onclick="geklikt();" />
</body>
</html>
```

Indien de alert opdracht niet in een functie was opgenomen, zou de alert opdracht uitgevoerd worden na het laden van de webpagina. Nu wordt de alert opdracht pas uitgevoerd als de gebruiker op de knop klikt. M.a.w. De opdracht binnen de functie geklikt() wordt pas na het klikken op de knop uitgevoerd.

De return opdracht

De return opdracht wordt gebruikt om waarden binnen een functie door te geven aan de opdracht die de functie gebruikt. Het voorbeeld berekent het product van twee getallen en geeft het resultaat terug:

```
<script>
function product(a, b) {
  return a * b;
 }
</script>
<p>4 x 3 = <span id="product"></span></p>
<script type="text/javascript">
 document.getElementById("product").innerHTML = product(4, 3);
</script>
```

Lokale variabelen

Als je binnen een functie een variabele met de opdracht var definieert, kan de variabele enkel in die functie gebruikt worden. Bij het verlaten van de functie wordt zo'n variabele vernietigd. Deze variabelen noemen we lokale variabelen. Je kunt lokale variabelen met dezelfde naam in verschillende functies definiëren en gebruiken, want ze worden enkel herkend binnen de functie waarin ze aangemaakt werden.

Bij het definiëren van een variabele buiten een functie, kan je de variabele binnen alle functies op een webpagina gebruiken. Zo'n variabele is bruikbaar vanaf het aanmaken tot het verlaten (sluiten) van de webpagina. Dit noemen we globale variabelen.

Functie bibliotheken

Om functies op meerdere pagina's te kunnen gebruiken, plaats je ze in een afzonderlijk bestand. Dit bestand kan dan door verschillende pagina's geladen en gebruikt worden. Het voorbeeld bestaat dan uit het JavaScript bestand functies.js:

```
// JavaScript Document
function geklikt(){
 alert("Je hebt op de knop geklikt.");
}
function product(a, b){
 return a * b;
}
```

en een HTML pagina waarop de functies geladen en gebruikt worden:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Functies</title>
  <!-- Hier worden de functies geladen -->
  <script src="functies.js"></script>
</head>
<body>
  <!-- Hier wordt de functie geklikt gebruikt -->
  <input type="button" value="Klik hier" onclick="geklikt();" />
  <p>4 x 3 = <span id="product"></span></p>
  <script type="text/javascript">
    // Hier wordt de functie product gebruikt
    document.getElementById("product").innerHTML = product(4, 3);
  </script>
</body>
</html>
```

Opdrachten functies

De grootste

Maak een HTML pagina met twee tekstvelden in één alinea naast elkaar.
Plaats daaronder een alinea met een knop met de tekst Bereken.
Plaats daaronder een alinea met een tekstveld met de tag Grootste getal: (zie afbeelding).

Schrijf een functie die uit twee getallen het grootste getal teruggeeft.

Zorg dat bij het klikken op de knop Bereken een functie wordt uitgevoerd die:

- ➔ Het getal in het eerste tekstveld in een variabele plaatst.
- ➔ Het getal in het tweede tekstveld in een variabele plaatst.
- ➔ Het laatste tekstveld toont het grootste getal. Maak daarvoor gebruik van de eerder gemaakte functie om het grootste getal te zoeken.

De grootste van drie

Pas de HTML pagina aan zodat je een derde getal kunt ingeven en daarvan het grootste getal kunt opzoeken.

Klok

Een functie kan je ook aanroepen na een bepaalde tijd. We maken een klok.
Maak een pagina met voor de klok een aanpasbare alinea.
Schrijf de functie startKlok() die:

- ➔ De datum van vandaag in een variabele plaats.
- ➔ Het uur uit de variabele met de datum haalt en in een variabele plaatst (.getHours()).
- ➔ De minuten uit de variabele met de datum haalt en in een variabele plaatst (.getMinutes()).
- ➔ De seconden uit de variabele met de datum haalt en in een variabele plaatst (.getSeconds()).
- ➔ Stel met behulp van deze variabelen de tijd met de notatie uu:mm:ss samen en plaats deze in de aanpasbare alinea.
- ➔ Laat deze functie nogmaals na een halve seconde uitvoeren met de opdracht:
- ➔

```
let myTime = setTimeout('startClock()',500);
```

setTimeout is de opdracht die in het voorbeeld de functie startKlok() na 500 milliseconden (halve seconde) laat uitvoeren. Daar deze opdracht deel uitmaakt van deze functie wordt de functie tot het afsluiten van de webpagina om de halve seconde uitgevoerd.
Nu rest ons alleen nog de functie te starten na het laden van de webpagina. Dit kan door de body tag met een onload event uit te breiden:

```
<body onload="startClock()">
```

Minuten en seconden met twee cijfers weergeven

Bij het weergeven van de minuten en seconden worden getallen kleiner dan 10 met één cijfer weergegeven. Het weergeven van steeds twee cijfers is mooier.
Schrijf een functie waaraan je een getal doorgeeft en die getallen kleiner dan 10 laat voorafgaan door een 0 (m.a.w. 2 wordt 02 en 12 blijft 12) en de twee cijfers teruggeeft.
Gebruik deze functie om de minuten en seconden steeds met twee cijfers weer te geven.

JavaScript Lussen

Vaak wil je een stuk code verschillende keren na elkaar uitvoeren. In plaats van die gelijkaardige opdrachtregels telkens te herhalen, plaatsen we die in een lus.
JavaScript kent verschillende lussen:

- ➔ **for:** herhaalt een vast aantal keer een codeblok.
- ➔ **while**: herhaalt een codeblok tot aan een bepaalde voorwaarde is voldaan.

De for lus

De for lus wordt gebruikt als je op voorhand weet hoeveel keer je een stuk code wilt uitvoeren.

Syntaxis:

```
for (variabele = startwaarde; variabele <= eindwaarde; variabele = variabele + toename) {
  Uit te voeren code
}
```

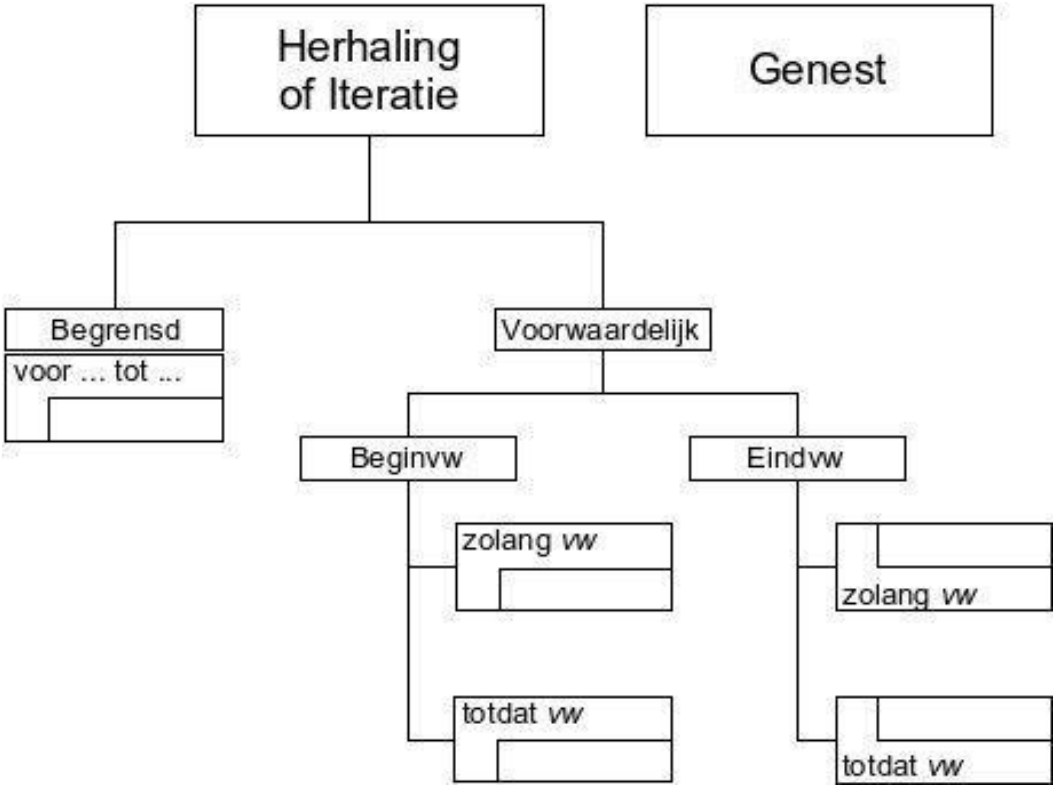
Voorbeeld (het weergeven van een voorbeeld van alle mogelijke koptags):

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Koppen</title>
</head>
<body>
  <div id="headings"></div>
  <script type="text/javascript">
    let myHeadings = "";
    for (kop = 1; kop <= 6; kop++){
      koppen += "<h" + kop + ">Dit is een Kop " + kop + "</h" + kop + ">";
    }
    document.getElementById('headings').innerHTML = koppen;
  </script>
</body>
</html>
```

Het voorbeeld gebruikt een lus waarbij in het begin de variabele kop de waarde 1 krijgt. De lus blijft lopen zolang de variabele kop een waarde heeft die kleiner of gelijk is aan zes. Na het voltooien van elke lus wordt bij de variabele blok één bijgeteld.

Opmerkingen:

- ➔ Na het uitvoeren van een codeblok kan je de waarde van de teller (kop) ook verlagen.
- ➔ De vergelijkingsoperator (<=) kan door elke ander vergelijkingsoperator vervangen worden. In het voorbeeld kan je de voorwaarde zonder het resultaat te beïnvloeden, vervangen door kop < 7.



De while lus

De while lus voert het codeblok uit tot aan de voorwaarde wordt voldaan.

Syntaxis:

```
while (variabele <= eindwaarde){ Uit
  te voeren code
}
```

Voorbeeld:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Koppen</title>
</head>
<body>
  <div id="koppen"></div>
  <script type="text/javascript">
    let koppen = "";
    let kop = 1;
    while (kop <= 6){
      koppen += "<h" + kop + ">Dit is een Kop " + kop + "</h" + kop + ">";
      kop++;
    }
    document.getElementById('koppen').innerHTML = koppen;
  </script>
</body>
</html>
```

De variabele kop krijgt de beginwaarde 1. Het codeblok wordt uitgevoerd zolang de waarde van kop kleiner of gelijk is aan zes. Op het einde van het codeblok wordt de variabele kop met één opgehoogd.

De do ... while lus

De do ... while lus is een variant op de while lus. Deze lus doorloopt het codeblok minstens één keer, en wordt herhaald zolang niet aan de voorwaarde wordt voldaan.

Syntaxis:

```
do {
  Uit te voeren code
} while (variabele <= eindwaarde);
```

Voorbeeld:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Koppen</title>
</head>
<body>
  <div id="koppen"></div>
  <script type="text/javascript">
    let koppen = "";
    let kop = 1;
    do {
      koppen += "<h" + kop + ">Dit is een Kop " + kop + "</h" + kop + ">";
      kop++;
    } while (kop <= 6);
    document.getElementById('koppen').innerHTML = koppen;
  </script>
</body>
</html>
```

Zelfs indien niet aan de voorwaarde wordt voldaan wordt de luscode éénmaal uitgevoerd. De voorwaarde wordt namelijk pas op het einde van de lus geëvalueerd.
(zie ook try / catch)

Onderbreken en doorgaan

De break opdracht

De break opdracht onderbreekt de lus en de aanwezige opdrachten na de lus worden uitgevoerd.

Voorbeeld:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Koppen</title>
</head>
<body>
  <div id="koppen"></div>
  <script>
    let koppen = "";
    for (kop = 1; kop <= 6; kop++){
      if(kop == 3){
            break;
      }
      koppen += "<h" + kop + ">Dit is een Kop " + kop + "</h" + kop + ">";
    }
    document.getElementById('koppen').innerHTML = koppen;
  </script>
</body>
</html>
```

Het voorbeeld toont dus enkel voorbeelden van Kop 1 en 2.

De continue opdracht

De continue opdracht onderbreekt de lus maar gaat verder met de volgende lusdoorloop.

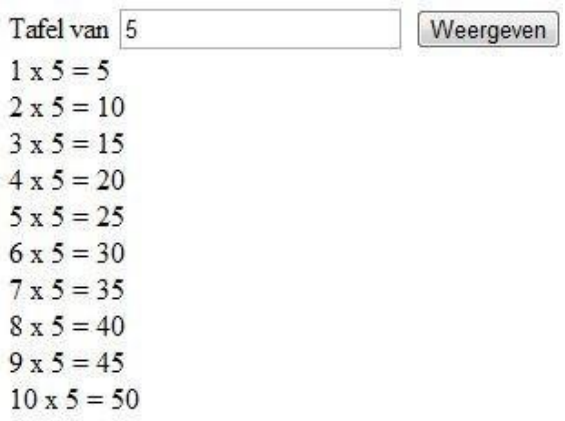
Voorbeeld:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Koppen</title>
</head>
<body>
  <div id="koppen"></div>
  <script type="text/javascript">
    let koppen = "";
    for (kop = 1; kop <= 6; kop++){
      if(kop == 3){
            continue;
      }
      koppen += "<h" + kop + ">Dit is een Kop " + kop + "</h" + kop + ">";
    }
    document.getElementById('koppen').innerHTML = koppen;
  </script>
</body>
</html>
```

Het voorbeeld toont alle Kop voorbeelden, uitgezonderd het voorbeeld voor Kop

Opdrachten lussen

Tafels van vermenigvuldiging



Maak een HTML pagina met een tekstveld met het id tafelvan en de tag Tafel van.

Plaats achter het tekstveld een knop met de tekst Weergeven. Bij het klikken op de knop Weergeven moet de de functie tafel('tafelvan') uitgevoerd worden.
Plaats daaronder een aanpasbare div tag.

Schrijf een functie waarmee je de tafels van vermenigvuldiging van een getal in een tekstveld met het opgegeven id in de aanpasbare div kunt weergeven. M.a.w.:

- ➔ Plaats met de opdracht parseInt(document.getElementById(getalid).value) het getal waarvan je de tafel van vermenigvuldiging moet weergeven in een variabele. Daarbij bevat getalid het id tafelvan die je bij het onclick event doorgaf.
- ➔ Plaats de volledige tafel van vermenigvuldiging in een variabele (de tag om een nieuwe regel te beginnen is
).
- ➔ Plaats de tafel van vermenigvuldiging in de aanpasbare div tag.

De functie universeler maken

Pas de functie aan zodat je ook het id van de tag waarin de tafel van vermenigvuldiging wordt weergegeven kunt meegeven. Nu de functie universeler inzetbaar is, plaats je de functie in een afzonderlijk JavaScript bestand.

JavaScript Events

Events zijn gebeurtenissen die door JavaScript opgemerkt worden, waardoor bij het optreden van zo'n gebeurtenis JavaScript code kan uitvoeren. M.a.w. JavaScript reageert op gebeurtenissen (events) met een actie (event handle). De events worden in de HTML tags geplaatst. (zie ook clickclickclick)

Voorbeelden van events:

- ➔ een muisklik
- ➔ een pagina of afbeelding laden
- ➔ een onderdeel met de muis aanwijzen
- ➔ een veld in een formulier selecteren
- ➔ een formulier verzenden
- ➔ een toets indrukken

Opmerking: Events worden meestal in combinatie met functies gebruikt. De functie wordt dan pas uitgevoerd na het voorkomen van de gebeurtenis (event).

De belangrijkste events

onLoad en onUnload

De onLoad en onUnload zijn gebeurtenissen die voorkomen bij het betreden en verlaten van een webpagina. De onLoad gebeurtenis kan gebruikt worden om het gebruikte browsertype en browserversie te achterhalen en daar gepast op te reageren.

De onLoad en onUnload gebeurtenissen worden vaak gebruikt in combinatie met cookies die aangemaakt worden bij het betreden of verlaten van een pagina. Voorbeeld: bij het eerste bezoek aan een webpagina kan naar de gewenste taal gevraagd worden. Na het opgeven wordt de gewenste taal in een cookie opgeslagen.

onFocus, onBlur en onChange

De onFocus, onBlur en onChange gebeurtenissen worden veel gebruikt voor het valideren van formuliervelden.

De onFocus event wordt actief als de gebruiker een formulierveld activeert om het in te vullen. De onBlur event wordt actief als ge gebruiker een formulierveldverlaat om een ander formulierveld te activeren.

Het voorbeeld maakt gebruik van de onChange event. De functie controleerEmail() wordt uitgevoerd bij het aanpassen van het tekstveld.

```
<input type="text" size="30" id="email" onchange="controleerEmail()" />
```

onSubmit

De onSubmit gebeurtenis wordt gebruikt om alle formuliervelden voor het verzenden te valideren. Het voorbeeld toont de werking van het onSubmit event. De functie controleerFormulier wordt uitgevoerd als de gebruiker op de knop Verzenden van het formulier klikt. Als de formuliervelden niet correct zijn ingevuld, gaat het verzenden niet door.

De functie controleerFormulier() geeft true (alles in orde) of false (niet alles in orde) terug. Bij true wordt het formulier verzonden, bij false wordt het verzenden afgebroken en kan de gebruiker zijn invoer corrigeren.

```
<form method="post" action="form.php" onsubmit="return controleerFormulier()">
```

onMouseOver en onMouseOut

Bij het aanwijzen van een HTML onderdeel ontvangt JavaScript een onMouseOver gebeurtenis.

Bij het verlaten van een aangewezen HTML onderdeel ontvangt JavaScript een onMouseOut gebeurtenis. Deze laatste gebeurtenis wordt veel gebruikt om een onMouseOver actie terug ongedaan te maken.

Voorbeeld: Bij het aanwijzen van een knop wordt de achtergrondkleur aangepast (onMouseOver). Bij het met de muiswijzer verlaten van de knop wordt de oorspronkelijke achtergrondkleur terug hersteld (onMouseOut).

Opdrachten events

In- en uitfaden

Maak een HTML pagina met een Kop 1 (h1 tag) met de tekst: De inhoud van deze pagina wordt langzaam zichtbaar. Plaats daaronder een koppeling (hyperlink, a tag) met de tekst Pagina verlaten en als koppeling # (koppeling die eigenlijk niets doet).

Maak de body tag met een CSS stijl (style="opacity:0") doorzichtig.

Zorg dat de pagina (body) na het laden een functie start om de inhoud zichtbaar te maken.
De functie heeft de volgende kenmerken:

- ➔ de transparantie van de body tag kan je opvragen of aanpassen met de opdracht
- ➔ document.body.style.opacity
- ➔ De body tag is volledig doorzichtig als de opacity stijl nul is, en volledig zichtbaar als de opacity één is, half doorzichtig bij een opacity van 0,5.
- ➔ als de pagina (body tag) volledig transparant is, moet de opacity groter worden tot de pagina volledig zichtbaar is
- ➔ als de pagina (body tag) volledig zichtbaar is, moet de opacity kleiner worden tot de pagina volledig doorzichtig is.
- ➔ de opacity wordt om de 0,1 seconde aangepast in stappen van 0,05.

Uitfaden bij het klikken op een koppeling

Zorg dat bij het klikken op de koppeling (a tag) een functie uitgevoerd wordt.
Deze functie heeft de volgende kenmerken:

- ➔ slaat de opgegeven webpagina waar je naar toe wilt surfen op in een variabele,
- ➔ voert de bovenstaande functie om de opacity aan te passen uit.

Pas de originele functie om de opacity aan te passen zo aan, dat bij het bereiken van een volledig doorzichtige pagina de opgegeven en opgeslagen webpagina geladen wordt. Om via JavaScript naar een pagina te surfen gebruik je de opdracht:

```
window.location.href = href;
```

waarbij de variabele href de URL van de webpagina bevat.