

Javascript – deel 07

Deze les behandelt eerst stringvergelijkingen en array sortering in Javascript. Daarna wordt uitgelegd hoe de data uit de verschillende form invoer elementen opgevraagd kan worden.

Maar eerst maken we een herhalingsopdracht!

Verslag

In dit deel van de cursus staan verschillende vragen die je moet beantwoorden en opdrachten om iets te maken of uit te proberen. Het is belangrijk dat je alle opdrachten zorgvuldig uitvoert!

Documenteer je werk in een verslag document 'javascript deel 07' waarin je

- voor elke uitprobeer opdracht een entry maakt met screenshots ter staving van wat je deed
- je antwoorden op de gestelde vragen neerschrijft

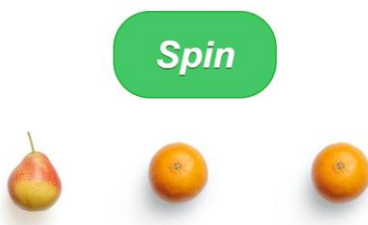
Oplossingen van grotere opdrachten (met veel code) bewaar je aparte folders in een Webstorm project.

Herhalingsopdracht slot machine

Deze herhalingsopdracht laat toe jezelf te testen in de materie van de vorige delen.

Download en unzip 'beginsituatie opdracht slot machine.zip' en vervolledig dit project met de volgende functionaliteit.

Initieel toont de webpagina dit (de afbeeldingen zijn willekeurig bepaald) :

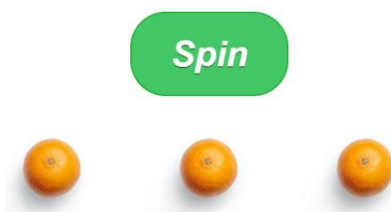


Telkens de gebruiker op de 'spin' knop klikt (die eigenlijk een hyperlink is!) kiest het programma drie willekeurige afbeeldingen en toont deze :



Zolang u niet stopt bent u geen verliezer!

Indien de gebruiker drie gelijke prentjes bekommt, is de boodschap onderaan anders :



Proficiat!

In de Javascript file vind je een globale variabele 'urls' die een array bevat met de urls van de vijf afbeeldingen.

Ze wijzen allen naar files in de /images folder, daar vind je dus de vijf afbeeldingen. Deze urls worden gebruikt als 'src' waarde van de elementen.

De HTML en CSS files bevatten al het nodige om de Spin button en de afbeeldingen te tonen.

Normaliter is het niet nodig de gegeven HTML elementen aan te passen, maar je zult wel zaken moeten toevoegen.

Bijvoorbeeld, om de berichten te tonen is er nog niks voorzien in de HTML of CSS code (de font is trouwens Arial 28px in het bold).

Tips :

- Om een willekeurige afbeelding te kiezen, bepaal je een willekeurige (random) getal en gebruikt dit als index in het 'urls' array
- Om de afbeeldingen te veranderen is het niet nodig de element te vervangen, geef ze gewoon een andere 'src' attribuut waarde
- Merk op dat de pagina initieel wél random afbeeldingen bevat maar geen bericht onderaan toont
- De element zelf hebben geen CSS-class, je zult dus iets creatiever moeten zijn met je selectoren om ze te pakken te krijgen. In de CSS file kun je zien hoe bv. de 'height' van de elementen wordt vastgelegd m.b.v. een selector `.slots img`
 - mocht het niet lukken, voeg dan gewoon een CSS-class toe aan de elementen

Strings vergelijken

Strings vergelijken doe je met `.localeCompare(andereTekst)`, bv.

```
let s1 = "peer";
let s2 = "appel";
let resultaat = s1.localeCompare(s2);
```

Deze method geeft een getal terug dat aangeeft welke tekst lexicografisch eerst komt :

- Het resultaat is negatief indien s1 voor s2 komt
- Het resultaat is positief indien s1 na s2 komt
- Het resultaat is 0 indien beide equivalent zijn

Aandachtspunten

- het is `localeCompare` niet `localCompare` (tussen de l en C staat nog een e)
- string vergelijkingen met `s1<s2`, `s1==s2`, `s1>s2` ?
 - kom je soms ook tegen, **maar zijn niet correct**
 - ze zijn gebaseerd op de ASCII code van de karakters, wat niet altijd het gewenste resultaat oplevert
 - ze werken blijkbaar niet in alle browsers op dezelfde manier
 - Zie <http://stackoverflow.com/questions/51165/how-do-you-do-string-comparison-in-javascript>

Testen op gelijkheid is in veel programmeertalen diepgaander dan je misschien zou verwachten. We kunnen immers op twee verschillende manieren vergelijken :

1. gaat het al dan niet om één en hetzelfde ding (**identity** of ook wel **fysieke gelijkheid**)
 - bv. verwijzen twee variabelen al dan niet naar hetzelfde object?
2. stellen twee dingen hetzelfde voor (**equality** of ook wel **logische gelijkheid**)
 - beschouwen we de objecten als gelijk of equivalent, op inhoudelijk vlak?

In javascript bestaat ook een opsplitsing, ze is terug te vinden in twee soorten vergelijkingsoperatoren

- `x == y`
- `x === y` ← doorgaans is dit degene die je wil gebruiken!

Helaas volgen deze twee operatoren niet altijd de identity vs. equality opdeling, in javascript zit het wat ingewikkelder in elkaar (afhankelijk van wat x en y precies zijn).

Voor strings is het echter heel gemakkelijk : de `==` en de `===` operatoren kijken allebei of de teksten in beide strings dezelfde inhoud hebben. Bij strings gaat het dus in beide gevallen om een equality check en nooit om een identity check!

Voor de meeste andere soorten waarden (numbers en strings, objecten, etc.) zijn er vele regels die het gedrag van == en === beschrijven. Ter illustratie kan je eens op volgende pagina kijken wat de problematiek zo ongeveer is

<http://stackoverflow.com/questions/359494>

We komen hier later misschien nog op terug.

Opdracht

Schrijf de nodige opdrachten op de console van de Chrome Developer Tools om aan te tonen dat strings met dezelfde inhoud altijd gelijk beschouwd worden volgens == en === ongeacht hoe je ze bekomt :

1. twee variabelen geïnitialiseerd met string literals

```
let s1 = "hallo";
let s2 = "hallo";
console.log( s1 == s2 );
console.log( s1 === s2 );
```
2. een string literal en het resultaat van een .slice() oproep

```
let s3 = "hallo";
let s4 = "hallo wereld!".slice(0,5);
console.log( s3 == s4 );
console.log( s3 === s4 );
```
3. het resultaat van een .slice() oproep en een string concatenatie

```
let s5 = "hallo wereld!".slice(0,5);
let s6 = "hal"+"lo";
console.log( s5 == s6 );
console.log( s5 === s6 );
```

Arrays sorteren

Arrays hebben ook een `sort()` functie om de elementen in het array te sorteren, zie

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

Zoals je in de documentatie kunt lezen, kunnen we aan `sort()` een optionele parameter meegeven die de vergelijkingsfunctie voorstelt waarmee elementen met elkaar moeten vergeleken worden.

Een functie verwijzing meegeven als parameter aan een andere functie is niks nieuws, we deden dit nml. ook al telkens we `.addEventListener()` gebruikten.

Die vergelijkingsfunctie vergelijkt twee elementen `a` en `b` met elkaar en geeft een getal terug om aan te duiden welk element het kleinste is:

resultaat moet negatief zijn indien $a < b$

resultaat moet 0 zijn indien $a = b$

resultaat moet positief zijn indien $a > b$

Merk op : de vergelijkingsfunctie is weliswaar optioneel, maar het standaardgedrag van `sort()` zonder vergelijkingsfunctie levert vaak een zinloze volgorde op. In de praktijk zul je dan ook bijna altijd een vergelijkingsfunctie voorzien.

Bijvoorbeeld, een array van getallen sorteren

```
const compare = (a,b) => {
    return a-b;
}
let array=[34, 67, 12, 5, 23];
array.sort(compare);
```

Je had misschien een ingewikkelde if-else verwacht? Bedenk dat de grootte van de return value er niet toe doet (bv. -13 is even goed als -42), enkel het teken is relevant (positief/negatief/nul). Kijk eens naar het teken van 'a-b' voor een aantal waarden voor `a` en `b`, je zult zien dat het telkens klopt!

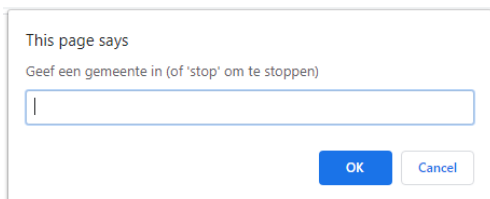
Bijvoorbeeld, een array van strings sorteren

```
const compare = (a, b) => {
    return a.localeCompare(b);
}
let array=["zebra", "aap", "giraf", "ezel"];
array.sort(compare);
```

Let erop hoe we strings vergelijken met `localeCompare` en niet met `<` of `>`, zoals eerder werd uitgelegd.

Opdracht Gemeenten

Schrijf een programma dat de gebruiker herhaaldelijk om een gemeente vraagt :



This page says

Geef een gemeente in (of 'stop' om te stoppen)

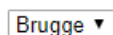
OK Cancel

Op die manier verzamelt het programma dus een lijst van gemeenten, totdat de gebruiker 'stop' ingeeft of op cancel klikt.

Het programma sorteert vervolgens deze lijst en gebruikt ze om een `<select>` element met `<option>` elementen op te vullen.

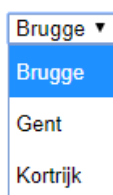
Voorzie hiervoor in je HTML document een `<select>` element met id 'selGemeenten' en voeg in je Javascript code, per gemeente een `<option>` element toe (als kind van het `<select>` element).

Bv. als de gebruiker achtereenvolgens Gent, Brugge en Kortrijk invoert verschijnt er :



Brugge ▼

Door op het driehoekje te klikken, klapt het `<select>` element open en zie je de gemeenten in alfabetische volgorde :



Brugge ▼

Brugge

Gent

Kortrijk

Dit openklappen is standaard gedrag van een `<select>` element, je hoeft dit dus niet zelf te programmeren!

Waarden in een form opvragen

In je HTML cursus heb je wellicht gezien dat er [allerlei <input> varianten](#) bestaan die je kunt gebruiken in een invulformulier.

Om de waarde van een input element op te vragen, kunnen we de **.value** property opvragen van het corresponderende DOM-tree element. Let er wel op dat dit steeds een string zal zijn, ook al heeft het <input> element bv. een type="number" attribuut (je zult dit dus nog moeten parsen)!

Bijvoorbeeld

In de HTML code

```
<input type="number" id="txtAmount">
```

In de Javascript code

```
let txtAmount = document.querySelector("#txtAmount");
let amountAsText = txtAmount.value;
let amount = Number.parseInt(amountAsText, 10);
```

Voor een **checkbox** (<input type="checkbox">) kun je de **.checked** property opvragen. Dit levert een boolean waarde (true/false) op, al naargelang of de checkbox is aangevinkt.

Bijvoorbeeld

In de HTML code

```
<input type="checkbox" id="chkPriority">
```

In de javascript code

```
let chkPriority = document.querySelector("#chkPriority");
console.log( chkPriority.checked );
```

Een **radiobutton** (<input type="radio">) heeft eveneens een boolean **.checked** property, waarmee je kunt achterhalen of de radiobutton aan is of niet. Radiobuttons behoren doorgaans tot een groep, nml. alle radiobuttons met eenzelfde 'name' attribuut.

In principe zou je dus alle radiobuttons in de groep moeten overlopen om te achterhalen welke 'aan' staat. Het kan echter ook veel eenvoudiger door een slimme CSS-selector te schrijven op basis van het 'name' attribuut en [de :checked pseudo-class](#) !

Bijvoorbeeld

In de HTML code

```
<input type="radio" name="colors" value="red" checked>
<input type="radio" name="colors" value="green">
<input type="radio" name="colors" value="blue">
```

In de javascript code

```
var checkedRadioButton = document.querySelector("input[name='colors']:checked")
```

Doorgaans is er exact 1 radiobutton geselecteerd, maar in principe is 0 ook mogelijk als je in je HTML code geen enkele radiobutton het 'checked' attribuut gaf!

De geselecteerde options in een **<select>** element opvragen is wat ingewikkelder. Een DOM-node voor een **<select>** element heeft volgende relevante properties :

- **.options**
een verzameling DOM-tree elements met de **<option>** elementen uit het **<select>** element. Deze kan op dezelfde manier als een array gebruikt worden (maar het is strikt genomen een NodeList).
- **.selectedIndex**
dit is de index (in bovenstaande .options lijst) van de eerste geselecteerde option, of -1 als er geen enkele geselecteerd is

De truuk is dus via de .selectedIndex de geselecteerde option opzoeken in de .options verzameling.

Het DOM-element van een **<option>** heeft volgende nuttige properties :

- **.selected**
een boolean die aangeeft of de option geselecteerd is
- **.value**
de waarde van het value attribuut van de option
- **.text**
de tekst van de option (tussen begin- en eindtag)

Indien een **<select>** meerdere geselecteerde options toelaat, zul je ze allemaal moeten overlopen en de waarde van hun .selected property nagaan!

Je kan je bij <select> elementen echter heel wat werk besparen door een slimmere CSS-selector te gebruiken!

Bijvoorbeeld :

In de HTML code

```
<select id="selfruit" size="4" multiple>
  <option>appel</option>
  <option>peer</option>
  <option>banaan</option>
  <option>kiwi</option>
</select>
```

In de javascript code

```
let selectedOptions=document.querySelectorAll("#selfruit option:checked");
```

De variabele selectedOptions zal nu een lijst bevatten met de geselecteerde options.

Als je er maar eentje verwacht omdat het <select> element geen multiple vermeldt, kun je natuurlijk querySelector() gebruiken i.p.v. querySelectorAll().

Opdracht formwaarden

Schrijf een HTML-pagina die er als volgt uitziet

Is roker ☐

Moedertaal ☐ Nederlands ☒ Frans ☐ Engels

Favoriete buurland

Bestelling

- aardappelen
- brood
- melk
- biefstuk
- chips
- krant

De invoermogelijkheden zijn

- Is roker : een checkbox.
- Moedertaal : een radiobutton groep met keuzes "Nederlands", "Frans" en "Engels" met resp. values "nl", "fr" en "en".
- Favoriete buurland : een enkelvoudige select met keuzes "Nederland", "Frankrijk", "Duitsland".
- Bestelling : een multi-select met keuzes "aardappelen", "brood", "melk", "biefstuk", "chips" en "krant".

Als er op de "Toon resultaat" knop geklikt wordt, verschijnen de uitgelezen waarden op de console.

Bijvoorbeeld, met bovenstaande keuzes zal de console de volgende output tonen :

```
is geen roker
moedertaal is nl
favoriete buurland is Frankrijk
bestelling bestaat uit aardappelen melk biefstuk
```