Javascript – deel 05

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.

Permanente evaluatie

In de volgende les kan je docent je oplossingen van de opdrachten opvragen en laten meetellen voor je permanente evaluatie.

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 05' 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.

Strings opdelen

Vaak is het nodig om een nieuwe String te bemachtigen die gebaseerd is op een stukje van een andere String. Bijvoorbeeld, als je in een tekst een zoekwoord wil vervangen door een vervangwoord moet je bij een treffer, de tekst voor en na die treffer gebruiken om de nieuwe tekst te bouwen.

Om zo'n deeltekst te verkrijgen heb je drie ingrediënten nodig: de String waarop je je baseert, de start indexpositie en tenslotte de eind indexpositie (of evt. het aantal karakters geteld vanaf de startpositie).

Om deelteksten uit een tekst te kunnen gebruiken, bestaan er in javascript maar liefst drie verschillende methods :

- .substring(idx1, idx2)
- .substr(idx, length)
- .slice(idx1, idx2) // dit is de betere keuze

De methods .slice() en .substring() werken min of meer op dezelfde manier, .substr() werkt echter iets anders. Zie onderstaande link voor een bespreking :

http://www.jacklmoore.com/notes/substring-substr-slice-javascript/

In je eigen code gebruik je liefst de .slice() method, die werkt altijd zoals je zou verwachten. Bijvoorbeeld

```
let s1="Hello world";
console.log( s1.slice(3, 8) ); // dit toont lo wo
```

Merk op dat het karakter 'r' op indexpositie 8 niet in de deeltekst zal zitten, de rechtergrens index is niet inclusief.

Opdracht : trigrams

Een trigram is een opeenvolging van drie letters die na elkaar voorkomen in een tekst. Bijvoorbeeld, in het woord "plezier" zitten volgende trigrams vervat :

ple
lez
ezi
zie
ier

Schrijf een programma dat alle trigrams van het woord "onoorbaar" achterhaalt en op de console zet.

-

Opdracht : de en het

Gegeven is de tekst "Gisteren zat de jongen op de stoep en at de helft van de appel". Schrijf een programma dat telkens 'de' vervangt door 'het' en het resultaat op de console zet. Je hoeft niet persé eerst de ganse zin om te zetten alvorens console output te produceren, je mag dit gaandeweg doen. Ga na dat je oplossing ook werkt voor een zin met 'de' vooraan en/of achteraan (bv. 'de man riep de').

-

Om letterlijke string waarden (i.e. string literals) in source code te plaatsen hebben we hierboven steeds de tekst tussen dubbele aanhalingstekens geplaatst, bv.

```
let tekst="deze tekst staat tussen dubbele aanhalingstekens";
```

Je mag echter ook enkele aanhalingstekens gebruiken :

```
let tekst='deze tekst staat tussen enkele aanhalingstekens';
```

Je kunt ook beide door elkaar mengen, dit is handig in de tekst zelf aanhalingstekens moet bevatten

```
let message= "U drukte op 'stop'.";
let message ='U drukte op "stop".';
```

Sommige speciale karakters moeten in een string literal door een backspace \ karakter voorafgegaan worden :

- \n newline
- \' single quote
- \" double quote
- \\ backslash
- \u89ab voor unicode karakter 89ab

Om de hoofdletter (of kleine letter) versie van een string te verkrijgen kun je de volgende twee methods gebruiken :

- .toUpperCase()
- .toLowerCase()

Het resultaat van deze methods is een nieuwe string met inhoudelijk dezelfde tekst als de originele string maar met enkel hoofdletters (of kleine letters).

Soms wil je zeker zijn dat een string geen lege ruimte (whitespace) bevat aan het begin of einde, dit is bv. handig om extra spaties of tab-karakters uit user-input te verwijderen. Je kunt hiervoor de .trim() method gebruiken :

```
let tekst="    Jan Janssens  ";
let proper=tekst.trim();                    // bevat "Jan Janssens"
```

Deze method produceert een nieuwe string waarin vooraan en achteraan geen whitespace meer voorkomt.

Strings vergelijken

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

```
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 local**e**Compare 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**)
2. stellen twee dingen hetzelfde voor (**equality**)

In java is de opdeling heel duidelijk. Indien a en b variabelen zijn van een reference type, dan is

1. a==b een identity vergelijking
   wijzen beide variabelen al dan niet naar hetzelfde object?
2. a.equals(b) een equality check
   beschouwt de auteur (van de .equals method van de relevante klasse) de objecten als gelijk?

Ook in javascript bestaat deze opsplitsing, ze is terug te vinden in twee soorten vergelijkingsoperatoren

- x == y
- x === y

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 geinitialiseerd met string literals
2. een string literal en het resultaat van een .slice() oproep
3. het resultaat van een .slice() oproep en een string concatenatie

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 dat de vergelijkingsfunctie weliswaar optioneel is, maar dat het standaardgedrag van sort() zonder vergelijkingsfunctie zelden sorteert zoals je zou verwachten!!

Bijvoorbeeld, een array van getallen sorteren

```
const compare = (a,b) => {
        return a-b; // enkel negatief, positie of nul is relevant voor return, dus zo kan het ook
}
let array=[34, 67, 12, 5, 23];
array.sort(compare);
```

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

Maak de opdracht gemeenten.

Waarden in een form opvragen

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.

Voor een checkbox kun je de .checked property opvragen (een boolean value).

Radiobuttons hebben eveneens een .checked property die je kunt bekijken; er zal er maar ene de waarde true hebben. Om alle radiobuttons in eenzelfde groep te bemachtigen (i.e. die zelfde name waarde hebben) kun je gebruik maken van de onderstaande document method :

```
document.getElementsByName("xyz")
```

Dit zal een NodeList opleveren van alle elementen met een attribuut name="xyz".

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 er geen).

- **.selectedIndex**
  de index in bovenstaande .options verzameling van de eerste geselecteerde option
  is -1 indien er geeneen 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.

Opdracht formwaarden

Schrijf een HTML-pagina die er als volgt uitziet

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
```

Opdracht formvalidatie

Maak een invulformulier voor de volgende gegevens :

- voornaam
  - tekst
  - mag leeg zijn
  - maximum 30 karakters (error : "max. 30 karakters")
- familienaam
  - tekst
  - mag niet leeg zijn (error : "verplicht veld")
  - maximum 50 karakters (error : "max 50 karakters")
- geboortedatum
  - datum
  - niet leeg (error : "verplicht veld")
  - ISO-formaat (2013-12-31) (error : "formaat is niet jjjj-mm-dd")
  - Voor de eenvoud controleren we enkel het formaat en niet of het ook daadwerkelijk een 'goeie' datum is (zie hieronder)
- email
  - tekst
  - niet leeg (error : "verplicht veld")
  - exact 1 @-teken (error : "geen geldig email adres")
  - min 1 karakter voor en na het @ teken (error : "geen geldig email adres")
  - (trouwens, zelf code schrijven om een email adres te valideren is een hopeloze opgave, zie bv. http://www.regular-expressions.info/email.html)
- aantal kinderen
  - getal (error : "is geen positief getal")
  - niet negatief (error : "is geen positief getal")
  - kleiner dan 99 (error : "is te vruchtbaar")

Voorzie voor elk veld ook een bijbehorend label en plaats alle velden onder elkaar.

Voorzie een knop "Valideer" die de inputvelden valideert (d.w.z. controleert) als je erop klikt :

- velden wiens waarde niet geldig is, krijgen een rood kader en rechts ervan komt de error mededeling in het rood
- velden wiens waarde wel geldig is, krijgen geen kader en hebben rechts geen foutmelding
- indien alle velden goed zijn ingevuld verschijnt er een popup met de melding 'proficiat!'

Extra lege ruimte links of rechts van ingevoerde tekst is steeds toegelaten.

In HTML5 kunnen we semantische input elementen maken voor allerlei gegevens, bv.

getal : type="number" of type="range"

datum : type="date"

email : type="email"

Normaliter zouden we hier voor deze form dankbaar gebruik van maken : alle controles die de browser voor ons doet, hoeven we immers zelf niet te schrijven.

Helaas ondersteunen sommige browserversies (die op dit moment nog relevant zijn) niet alle controles, zie bv. http://caniuse.com/#feat=input-email-tel-url en let op de entry voor IE9.

In de praktijk zul je eerder gebruik maken van een externe library om je forms aan de clientkant te valideren, bv. JQuery Validation Plugin of Parsley.js.

Bedenk trouwens dat je in het programma aan de serverkant hoedanook de controles moet uitvoeren, ongeacht of dit al in de browser gebeurde. Hackers kunnen immers makkelijk de controles in de browser omzeilen en zo niet-toegelaten data naar de server sturen.

In deze oefening beperken we ons opzettelijk tot input elementen met type="text" zodat we nog wat op strings kunnen oefenen.

Om dezelfde reden zullen we de ISO datum niet eenvoudig parsen via de Date.parse method, maar de jaar, maand en dag stukken 'manueel' uit de string halen en slechts beperkt controleren :

- het jaartal moet 4 karakters lang zijn en een getal voorstellen
- de maand en dag stukken moeten elk 2 karakters lang zijn en een positief getal voorstellen
- er moeten twee '-' streepjes in voorkomen op de juiste plaatsen

Deze controle is verre van waterdicht, bv. 2013-02-31 voldoet aan bovenstaande regels (en geeft dus geen problemen bij het valideren) maar is geen geldige datum.

Om na te gaan of een string een geldig getal bevat, kan je de volgende hulpfunctie gebruiken :

```
const isGetal = (tekst) => {
        return !isNaN(tekst);
}
```

Het filmpje 'opdracht formvalidatie' legt uit hoe je deze opdracht kunt aanpakken.