

Labo Gebruikersinterfaces

Reeks 2: JavaScript

Doel: Het formulier uit vorig labo interactief maken met Javascript.

1 Inleiding

Vorig labo maakte je een website met twee pagina's: de hoofdpagina (die draagt altijd de naam `index.html`) en een formulier. We passen dit formulier nu zo aan, dat het winkelmandje gevuld kan worden. Voor de betaling voorzien we ook code (weliswaar dummy; je zal er niet rijk van worden...).

2 Getting started

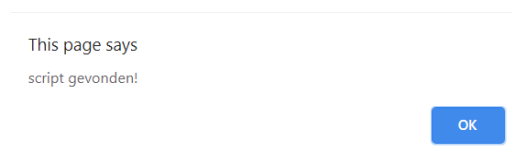
Tijdens dit labo schrijf je ES6 Javascript code (let variabelen, arrow functions, classes, modules). ES6 is vrij nieuw, om het correct werkend te krijgen moeten we nog een aantal opties aanpassen in Webstorm/IntelliJ:

- Configureer Webstorm zodat ES6 gebruikt wordt: File -> Settings -> Languages & Frameworks -> Javascript
- Vink "Allow unsigned requests" aan in File -> Settings -> Build, Execution, Deployment -> Debugger.

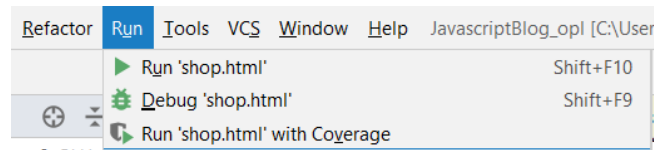
Werk verder in het project van vorige week. Voeg een map met naam `scripts` toe, en voeg daaraan een nieuw `JavaScript`-bestand met naam `shop.js` toe. In het `shop.html`-bestand zorg je voor de koppeling naar dit `.js`-bestand.

In het bestand `shop.js` schrijf je de nodige `JavaScript`-code. Controleer of het linken goed gebeurd is: als onderstaande regel code in `shop.js` staat, zou je een waarschuwing moeten krijgen bij het openen van de pagina.

```
alert("script gevonden!");
```



Let op: als je in de browser (bv Google Chrome) surft naar het bestand `shop.html` en je krijgt de gewenste popup niet te zien, kan je nagaan wat er aan de hand is met een klik rechts op de pagina: kies voor *'Inspect'*. Rechts krijg je dan verschillende tabbladen te zien waar je informatie over de html-code, css, achterliggende scripts, foutmeldingen,... kan aflezen. Zoek naar een rood signaal. Vind je een foutmelding die te maken heeft met **CORS**? Open het bestand `shop.html` dan niet vanuit de browser, maar run het bestand vanuit Webstorm.



3 Reactie op knoppen

Voeg een eventlistener toe aan de knop met opschrift `voeg toe aan winkelkar`. Ga in kleine stapjes te werk.

- Om een component van (het formulier in) het html-bestand aan te spreken, zoek je deze op aan de hand van zijn id.

```
let knop =document.getElementById("idKnopVoegToe");
```

Dit maakt meteen duidelijk waarom het nuttig is om id's in te stellen op componenten van het formulier.

- Let op als je de eventlistener instelt. Het rechterlid is de *naam* van een functie, niet de *functieoproep*. Er komen dus geen lege ronde haakjes na `voegToeAanWinkelkar`!

```
document.getElementById("idVoegToe").onclick = voegToeAanWinkelkar;
```

Merk op dat er alternatieven zijn voor bovenstaande code (maar dezelfde opmerking over de haakjes blijft).

- Om de staat van bepaalde componenten (checkboxes, radiobuttons,...) na te gaan bestaan er voldoende methodes. Gebruik de hulp die Webstorm biedt en/of gebruik en verfijn je zoek-skills (slides van de theorieles, je lesnota's, [documentatie van html](#), het internet).
- In het begin bereken je enkel de totale som van de bestelde goederen (neem abstractie van de meerprijs voor een houten kader). Deze totale som toon je in het tekstveld rechts. (Je mag hier een `textarea` van maken in plaats van platte tekst. Voeg ook bootstrap-classes toe aan dit tekstgebied.) Je mag de oude inhoud van het winkelwagentje ook telkens overschrijven (dus nog geen accumulatie).

4 Klassen Product en Winkelkar

Als het voorgaande werkt, willen we ervoor zorgen dat het winkelwagentje telkens bijgevuld kan worden. Daarvoor moeten we in het programma informatie onthouden. Schrijf twee klassen, die dit zullen toelaten.

- De klasse `Product` houdt de naam en de prijs van een product bij. Ook het aantal stuks dat van dit product besteld werd, mag je hier bijhouden (er zijn andere ontwerpkeuzes te maken; we houden het hier bewust eenvoudig). Om de output niet te lang te maken,

zullen we de drie producten uit de webshop volgende namen geven (dat is nog niet van belang voor het schrijven van de klasse, wel voor het latere gebruik):

tekst per mail
zeefdruk op papier
zeefdruk op papier, ingekaderd

Voorzie minstens een constructor, een methode `koop()` die één stuk extra van dit product aankoopt, een methode die de totale kostprijs van het gekochte aantal producten teruggeeft, en een methode om de informatie uit te schrijven (naam en totale kostprijs).

- De klasse `Winkelkar` bewaart een map, waarin de naam van een product afgebeeld wordt op een object van de klasse `Product`. Vul de map al op met de drie producten die in de webshop te koop aangeboden worden (dit mag hardgecodeerd zijn, later lossen we dit mooier op).

Voorzie een methode om één stuk van een bepaald product aan te kopen; de naam van het product wordt als parameter meegegeven.

Voorzie een methode die de totaalprijs van de winkelkar teruggeeft.

Voorzie een methode die de informatie uit de winkelkar netjes uitschrijft, inclusief totaalprijs.

Voorzie een methode die de winkelkar weer leeg maakt.

5 Gebruik van de klassen `Product` en `Winkelkar`

Gebruik nu deze klassen om de werking van de knoppen met opschrift `voeg toe aan winkelkar` en `Maak leeg` op punt te stellen. Een mogelijke weergave van de winkelkar vind je in de figuur hiernaast.

The screenshot shows a web form for a shopping cart. At the top, there's a header 'winkelkar'. Below it, a list of items is displayed: '1x tekst per mail (0.5)', '5x zeefdruk op papier (50)', and '2x zeefdruk op papier, ingekaderd (80)'. The total price is shown as 'TOT € 130.5'. Below the list, there's a section for payment method, 'betaling per', with two radio buttons: 'overschrijving' (selected) and 'betaalkaart'. At the bottom, there are two buttons: 'Maak leeg' and 'Bestel'.

6 Controle vóór het insturen van een formulier

In de html-code had je allicht al code geschreven om via de knop `Bestel` de bestelling door te sturen. Het is echter wijzer om eerst te controleren of de `submit`-actie wel door mag gaan.

Zoek eerst informatie op over hoe een functie die de controle zal uitvoeren, gelinkt moet worden aan het formulier. (Hierbij is al verkapt dat de controle gelinkt wordt aan het formulier, en niet aan de button met opschrift **Bestel**.) Neem hier gerust je tijd voor; probeer verschillende zoektermen en leid de volgende zoektermen af uit het resultaat van de eerste zoekopdrachten.

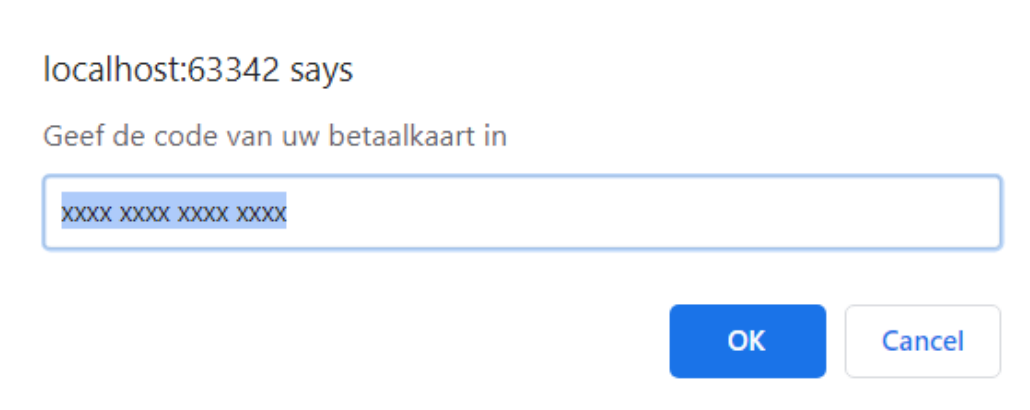
We geven nog deze tip: je zal allicht verschillende oplossingen vinden. Stel je niet tevreden met een oplossing waarin JavaScript-code in het html-document geschreven moet worden. Zoek in dat geval verder.

De functie die de controle uitvoert kan je voorlopig *schetsen*. Dat betekent dat je de functie een inhoud geeft die geen berekeningen doet en geen rekening houdt met eventuele parameters; ze mag behoorlijk hardgecodeerd zijn. De (eventuele) returnwaarde codeer je ook hard. Een voorbeeld:

```
function voerControleUit(){
    alert("controle wordt uitgevoerd");
    return 17;           // zomaar - omdat 17 een mooi getal is
}
```

7 Controle op input van tekst en getallen

Nu is het tijd om de controle ook effectief inhoud te geven. Indien de gebruiker aangeduid heeft dat hij per overschrijving betaalt, dan hoeft er niets te gebeuren. Indien de gebruiker met een betaalkaart wil afrekenen, dan laat je een popup verschijnen die een kaartnummer opvraagt.



De code van de meeste betaalkaarten kan gecontroleerd worden aan de hand van het algoritme van Luhn. In de cursus Informatica van 1e bachelor kregen jullie deze oefening al voorgeschoteld in Python. We herhalen kort.

Het algoritme van Luhn werkt van rechts naar links, waarbij het cijfer helemaal achteraan een controlecijfer is. Het controlecijfer moet de volgende test doorstaan.

- Startend vanaf het cijfer dat links van het controlecijfer staat, schuiven we telkens twee cijfers op naar links. We verdubbelen telkens de waarde van elk van deze cijfers. Als deze verdubbeling resulteert in een getal van twee cijfers, dan tellen we de cijfers van dit getal bij elkaar op. Op die manier wordt het cijfer 7 bijvoorbeeld verdubbeld tot 14, waarna de som van de cijfers gelijk is aan 5.
- Daarna nemen we de som van de cijfers van het kaartnummer, inclusief het controlecijfer.
- Als de totale som deelbaar is door tien, dan is kaartnummer geldig. Anders is het kaartnummer ongeldig.

Volgens dit algoritme is het kaartnummer 49927398716 dus geldig. De som wordt immers als volgt berekend

$$6 + (2) + 7 + (1 + 6) + 9 + (6) + 7 + (4) + 9 + (1 + 8) + 4 = 70$$

en dat resultaat is deelbaar door 10. Hierbij wordt de som van de cijfers na de verdubbeling aangegeven tussen ronde haakjes.

Indien de gebruiker een kaartnummer ingeeft dat niet voldoet aan deze test, dan wordt het formulier niet verwerkt.

Een paar tips om de controle uit te voeren.

- Schrijf al zeker een (hulp)functie `cijfersom` die de cijfersom van een getal bepaalt. Dit moet werken voor alle positieve gehele getallen, ongeacht uit hoeveel cijfers ze bestaan. Test uit.
- Vervang alle spaties door lege tekst. Let op, gebruik de `replace`-methode zoals het in JavaScript hoort: er is geen lus nodig om alle spaties weg te krijgen, wel de juiste parameters.
- Om na te gaan of alle karakters cijfers voorstellen, kan je de methode `includes` gebruiken: `"0123456789".includes(c)` met `c` één karakterteken. Zet de code eerst om naar een array van karakters (gebruik de spread operator) en gebruik zo mogelijk functie-uitdrukkingen (function expressions) in plaats van lussen om de karakters in de tekst te overlopen.

Tip Heb je tussendoor genoeg breakpoints gezet om je code te kunnen debuggen?

8 Wijzigen van html-DOM-structuur vanuit JavaScript-code

We hebben tot nu toe informatie uit het html-document gehaald door elementen uit de DOM-structuur op te roepen. Nu veranderen we de DOM-structuur van het html-document: we voegen een inputveld toe.

Indien de gebruiker een bestelling doet die per post opgestuurd wordt, vragen we naar zijn adres (we beperken ons tot één nieuw invulveld).

te bestellen	uw gegevens
<input type="checkbox"/> volledige tekst van 'Lorem Ipsum' toegestuurd per mail (€ 0.50)	naam <input type="text"/>
<input type="checkbox"/> volledige tekst, zeefdruk op handgeschept papier (€ 10.00)	postcode <input type="text"/>
<input checked="" type="checkbox"/> volledige tekst, zeefdruk op handgeschept papier, ingekaderd (€ 40.00); kleur kader <input type="text" value="wit"/>	straat en nr <input type="text"/>

Met andere woorden: als de gebruiker een vinkje zet in de tweede en/of derde checkbox, verschijnt er een extra invulveld na de postcode. Indien de gebruiker de vinkjes weer weghaalt, verdwijnt dat invulveld ook weer.

Let op Omdat het invulveld genest zit in een paar andere tags (die met Bootstrap te maken hebben), zal het niet volstaan om één element aan de DOM-structuur toe te voegen. Je zou dit kunnen omzeilen door letterlijke html-code toe te voegen door het attribuut `innerHTML` van een uitgebreid stuk html-code te voorzien. Dit is echter geen goed idee. Gebruik de zoektermen *innerHTML security considerations* of surf rechtstreeks naar developer.mozilla.org voor meer uitleg.

Extra Kan je ervoor zorgen dat de straat ingevuld *moet* zijn voor je iets aan het winkelmandje kan toevoegen?

Extra Stel dat de gebruiker een bestelling toevoegt aan het winkelmandje waarvoor zijn straat gekend dient te zijn. Als hij nadien opnieuw begint en het invulveld voor de straat verdwijnt, dan zou hij in principe een andere straat kunnen invullen. Kan je ervoor zorgen dat de gegevens van de eerste straat bewaard worden, zodat ze niet meer aangepast kunnen worden? Wat als het winkelmandje leeg gemaakt wordt?