

Labo Gebruikersinterfaces

Reeks 1: JavaScript

Doel

- Begrijpen hoe webpagina's *responsive* gemaakt worden met het *grid system* van Bootstrap.
- Webpagina interactief maken met behulp van JavaScript (eerst gegevens opvragen en html-DOM-structuur aanpassen; daarna ook gebruik van klassen, rekenen met getallen, cijfers en arrays in JavaScript).

Voorkennis

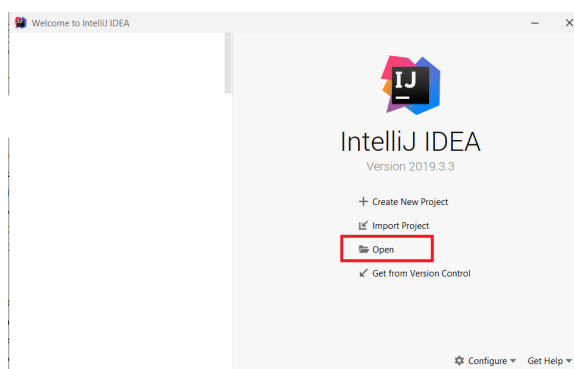
- Basis HTML en CSS.

1 Getting started

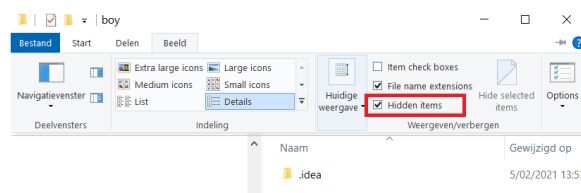
Er bestaan veel editors die je kan gebruiken om een HTML5-applicatie te maken. Tijdens deze labo's ben je in principe vrij in je keuze maar wij raden de [Ultimate edition van IntelliJ](#) aan. Om deze op je eigen toestel te installeren kan je een [gratis student license](#) aanvragen. Tijdens de test zal IntelliJ gebruikt worden op de computers van UGent.



In latere labo's wordt er verder gewerkt op een bestaand project, maar hier geven we er de voorkeur aan om de weg naar een nieuw project te tonen. Dat komt van pas als je zelf aan het experimenteren gaat.

1. Open een (bestands-)verkenner en maak een nieuwe map met naam *boy* aan. Zet deze meteen op de juiste plaats binnen je eigen documentstructuur. Deze map zal straks het project bevatten. Gebruik geen spaties in de naam van je map, dat is om problemen vragen.
2. Open IntelliJ, en selecteer *Open*.

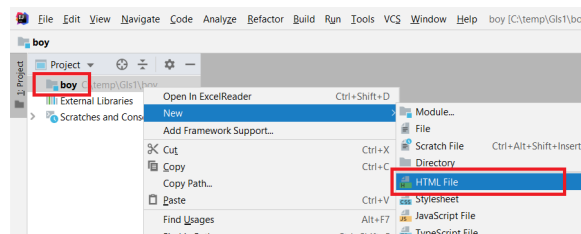


Navigeer dan naar de map die je zonet aanmaakte. Door deze map in IntelliJ te openen, zal er automatisch een onzichtbare submap met naam *.idea* aangemaakt worden. Dit kan je controleren als je in de verkenner in de tab *Beeld* het vakje *Hidden items* aanvinkt (moest dat nog niet gebeurd zijn).

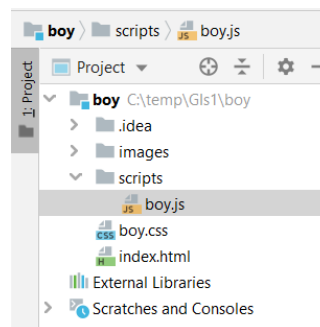


Nu kunnen we met recht en reden de map *boy* een *projectmap* noemen. Merk op: de volgende keer dat je deze map met IntelliJ opent, zal het icoontje aanduiden dat dit een projectmap is: je vindt  in plaats van .

3. Belangrijke tip: verander de naam van een projectmap niet zomaar. Het kan zijn dat het project dan niet meer naar behoren werkt.
4. Klik dan in IntelliJ rechts op de map *boy* en voeg een HTML(5) bestand toe; wij gaven het de naam *index*.



Vervolgens voeg je nog drie zaken toe: een css-bestand *boy.css*, een map (directory) *scripts* en daarin een javascript-bestand *boy.js*.



5. De startcode van het html-bestand en het css-bestand krijg je: download ze van Ufora en kopieer (ofwel kopieer je de inhoud, ofwel vervang je in een verkenner jouw bestanden door de nieuwe).
6. Bestudeer *index.html*, er worden vier zaken geïncludeerd: de bootstrap CSS, JQuery en Popper.js. JQuery is een Javascript bibliotheek die Bootstrap nodig heeft voor een deel van de functionaliteit. Popper is een library die gebruikt wordt door Bootstrap voor tooltips en popovers.

2 Bootstrap uitgelegd

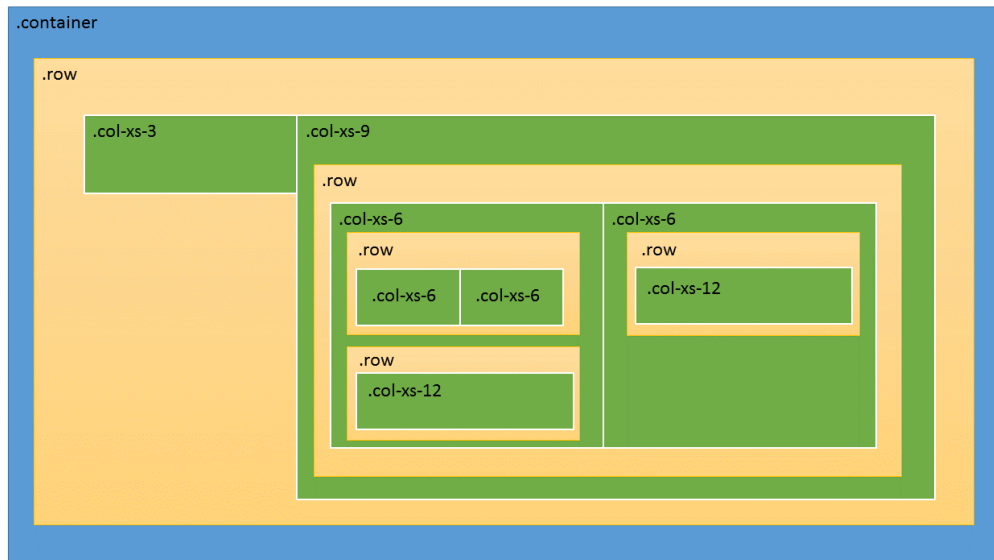
De inhoud en functionaliteit van een website is natuurlijk het allerbelangrijkste maar de lay-out en opmaak maken vaak het verschil tussen een webapplicatie die effectief gebruikt wordt en één die een stille dood sterft. Webdesign is een specialiteit op zich en het ontwerpen van een aantrekkelijke stijl behoort meestal niet tot de skill set van de programmeur die de website ook effectief implementeert.

Bootstrap is een front-end framework dat webdevelopers toelaat om het ontwikkelproces een pak te versnellen. Bootstrap bevat onder andere kant-en-klare CSS-klassen en Javascriptcode om snel een visueel aantrekkelijke website te maken. Bootstrap is echter meer dan een vereidelde template. Eén van de belangrijkste redenen om Bootstrap te gebruiken, is dat websites tegenwoordig op allerlei devices bekeken worden: van de (hele) grote schermen van een vaste computer tot het mini-schermpje van een smartphone. Als een website altijd dezelfde statische layout heeft, zullen deze schermen niet optimaal benut worden. Daarom is het goed om websites *responsive* te maken, dat wil zeggen dat ze zich automatisch aanpassen aan de drager waarop ze gelezen worden. Breek je een website op in verschillende rijen en kolommen, dan kunnen deze delen zo herschikt worden dat ze het aangeboden scherm maximaal benutten. We spreken hier echter niet van een tabel (die hoort altijd netjes samen te blijven zoals hij bedoeld is). In Bootstrap-terminologie spreken we van een *grid system*, met dynamisch herschaalbare kolommen.

Het gebruik van het grid system is vrij eenvoudig als je rekening houdt met volgende regels:

- De pagina bevat rijen en elke rij bevat kolommen; de inhoud komt in de gevormde cellen te staan.
- Er zijn steeds *12 gelijke basiskolommen per rij* (maar je kan kolommen samenvoegen om bredere kolommen te bekomen).
- Alle rijen moeten in een container-element geplaatst worden (html-element met klasse “container”).
- De breedte van een uiteindelijke kolom wordt ingesteld door aan te geven hoeveel van de 12 basiskolommen ze moet bevatten.
- De breedte van een uiteindelijke kolom kan voor verschillende schermgroottes anders ingesteld worden.
- Er kan ongelimiteerd genest worden.

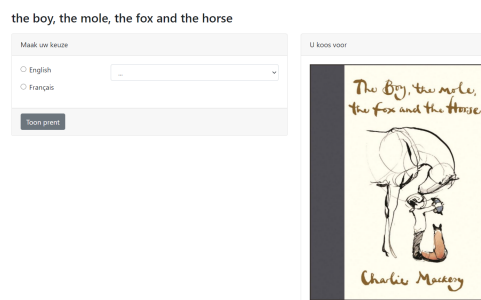
Het instellen van de breedte van een kolom gebeurt met behulp van CSS-klassen die gedefinieerd zijn door Bootstrap. De klassenaam bestaat uit twee delen: een aanduiding voor welke schermgrootte ze geldt en het aantal kolommen die ze moet bevatten. Onderstaande figuur illustreert dit.



De klasse `col-sm-6` bijvoorbeeld geeft aan dat dit element 6 kolommen moet omvatten bij kleine schermen en alles wat groter is. De mogelijke schermgroottes zijn `xs`, `sm`, `md`, `lg`, `xl` en `xxl`. Zo gebruik je `sm` voor tablets, `md` voor desktops en `lg` voor grotere desktops. Er bestaan ook “`col-*`” klassen die gelden voor alle schermgroottes. Meer informatie over het grid systeem kan je vinden op <https://getbootstrap.com/docs/5.1/layout/grid/>, of via de zoekwoorden `bootstrap grid`. Merk op: we gebruiken versie 5.1 van Bootstrap. Je hoeft deze documentatie niet nu onmiddellijk door te nemen, je moet wel weten dat je daar kan gaan zoeken.

3 Bootstrap toegepast

1. We keren terug naar het project *boy*. Open het html-bestand, en klik met de rechtermuis in dit bestand. Dan kan je halverwege het context-menu kiezen voor *Run*. De website zal openen in een browser (hopelijk Google Chrome). De site kan er uitzien zoals hieronder aangegeven, maar afhankelijk van de grootte van je scherm kunnen de blokken ook onder elkaar komen te staan. Verklein je scherm, en kijk wat er gebeurt. Of zoom in, dat heeft hetzelfde effect.



2. Zoals uitgelegd in bovenstaande paragraaf, is Bootstrap (o.a.) verantwoordelijk voor het *responsive* zijn van de site. Dit kan je nagaan door Bootstrap even uit te schakelen: zet bovenaan in het html-bestand de eerste `link`-tag in commentaar (tussen `<!--` en `-->`). Bewaar dit, en laat het html-bestand opnieuw runnen.
3. Haal de `link`-tag weer uit commentaar. Ga daarna zoeken in het css-bestand *boy.css*. Daar staat maar één selector voor een klasse vermeld: `.form-group`. Dat betekent dat alle andere klassen die in het bestand *index.html* gedeclareerd werden, al zeker tot het

Bootstrap-arsenaal behoren. Zoek een paar van deze klassen op in de [documentatie van Bootstrap](#):

- container (Klik op de link in de zin *See them in action and compare them in our Grid example.*, en speel met de grootte van het browser-venster: wat gebeurt er?)
- card-header (scroll vanuit het zoekresultaat naar boven om zicht te krijgen op de context)
- img-fluid (het zoekresultaat levert niet veel uitleg; leid hier wel een nieuw zoekwoord uit af dat meer resultaat levert)
- btn-secondary (zelfde opmerking)

Merk op: zelfs de klasse **form-group** is een Bootstrap-klasse. Alleen werd er in het lokale css-bestand *boy.css* nog wat code toegevoegd voor deze selector.

Nu heb je een idee wat mogelijk is met Bootstrap. Dit is uiteraard nog wat anders dan het effectief zelf gebruiken. Maar we concentreren ons voor het vervolg graag op het zelf gebruiken van JavaScript.

4 JavaScript

Nu wordt er gevraagd om ES6 JavaScript-code te schrijven in het bestand *boy.js*, met **let** variabelen, *arrow functions*, klassen en modules. Mogelijks moet je nog een aantal opties aanpassen in IntelliJ om dit werkende te krijgen. Zet deze stappen alleen als het nodig blijkt.

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

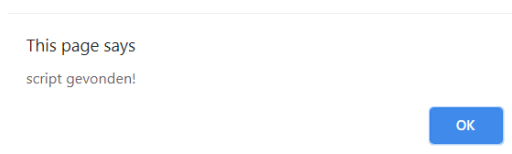
4.1 JavaScript linken

Om te beginnen leggen we de koppeling tussen het html-bestand en het js-bestand. Raadpleeg de theorie en zet de juiste code helemaal onderaan, tussen deze regels commentaar (zet die eerste regel commentaar er eventueel nog bij, als die er niet staat):

```
<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
```

In het bestand *boy.js* schrijf je straks de nodige JavaScript-code. Controleer nu alvast of het linken goed gebeurt is: als onderstaande regel code in *boy.js* staat, zou je een waarschuwing moeten krijgen bij het openen van de pagina.

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



4.2 JavaScript debuggen

Fouten die at runtime gedetecteerd worden, worden gemeld daar waar het programma loopt: in de browser, niet in de IDE. Alleen worden die foutmeldingen verstoep voor het ‘grote publiek’. Je kan ze enkel zien als je de pagina ‘*inspecteert*’.

Om dit te illustreren kan je een letter weglaten uit de functie-oproep `alert(...)`. Laat het html-bestand lopen vanuit IntelliJ. Dan zal de waarschuwing niet verschijnen, maar verder lijkt er niets aan de hand: de website wordt getoond. Klik nu rechts op de webpagina en 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, en klik om de meldingen te lezen. Herstel daarna de JavaScript-code.

Tot nu toe werd aangeraden om de website te openen vanuit IntelliJ. Probeer nu de website te openen door vanuit de browser naar het lokale bestand te surfen: tik het volledige pad in, zoals `C:/.../boy/index.html`. Of klik het bestand `index.html` vanuit een bestandsverkenner open. Krijg je de gewenste popup niet te zien? Inspecteer wat er aan de hand is. Allicht krijg je een foutmelding die te maken heeft met `CORS`. In dat geval open je de webpagina altijd vanuit IntelliJ.

5 Webpagina interactief maken

De website zoals nu voorgeschoteld, is nog niet interactief. De werking zou als volgt moeten gaan:

- Bij het aanvinken van een radio button, wordt de keuzelijst opgevuld met aangepaste termen. Voor *English* zijn dat de woorden **brave**, **home**, **strength** en **who**. Voor *Français* zijn dat de woorden **grand**, **mieux** en **petit**.
- De bezoeker van de site kan in de keuzelijst een keuze aanduiden - ogenschijnlijk gebeurt er dan nog niets.
- Als de bezoeker op de knop *Toon prent* klikt, zal de huidige afbeelding vervangen worden door de afbeelding die de bezoeker aanduidde in de keuzelijst. De afbeeldingen zelf zijn allemaal te vinden in de map `images` (die kreeg je in `boy.zip`). Zet deze map naast de map `scripts` in de projectmap. Doe dat in een verkenner, niet in IntelliJ. Je kan daarna in IntelliJ wel de bestanden zien staan. Merk op: de benamingen zijn logisch opgebouwd.

Om dit alles te kunnen realiseren, gaan we stap voor stap tewerk. Gebruik in ieder geval de hulp die IntelliJ biedt, en gebruik/verfijn je zoek-skills. Gebruik de slides van de theorieles, je lesnota's, [documentatie van html](#), het internet,...

5.1 Reactie op knop ‘Toon prent’: instellen van eventlistener

We beginnen met een eenvoudig stukje, dat toch al meteen effect toont. Bij een druk op de knop *Toon prent* zal de afbeelding uit het bestand `brave.png` getoond worden - voorlopig hardgecodeerd.

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

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

Indien de knop nog geen id heeft, voeg je die toe. (Je zou ook de methode-aanroep `getElementsByTagName("button")` kunnen gebruiken, maar dan moet je nog het eerste element uit deze lijst halen. En dan moet je ook zeker zijn dat er geen buttons meer bijkomen in het html-bestand... Tricky!)

2. 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 `toonPrent!`

```
document.getElementById("id_knop").onclick = toonPrent;
```

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

5.2 Reactie op aanklikken radiobutton: aanpassen van DOM-structuur

1. Bij het klikken op een radiobutton moet de keuzelijst gevuld worden. Omdat deze reactie (afgezien van de concrete items) gelijk is bij beide radiobuttons, implementeer je één methode `vulKeuzelijst` - zonder parameters. Deze methode zal nagaan welke radiobutton aangevinkt is, en zal aan de hand daarvan de parameters bepalen voor een hulpmethode `vulKeuzelijstMet(...)`.

Merk op: JavaScript kent geen overloading van methodes; declareer dus nooit methodes met dezelfde naam, ook niet als ze duidelijk een andere parameterlijst hebben.

2. De methode `vulKeuzelijstMet(...)` doet dan het echte werk. Er moet html-code ‘bijgeplakt’ worden in het html-bestand. Dit doen we door de *html-DOM-structuur* aan te passen. Om die structuur te kunnen aflezen, is het nuttig om de html-code voor ogen te houden:

```
<select class="form-control" name="prenten">
  <option value="grand">grand</option>
  <option value="mieux">mieux</option>
  <option value="petit">petit</option>
</select>
```

5.3 Kiezen uit de lijst: informatie bewaren

Tot nu toe hebben we 2/3e van de functionaliteit verzorgd: de radiobuttons reageren, en de knop *Toon prent* reageert (toegegeven, dat is nog hardgecodeerd...).

Nu moeten we nog rekening houden met het item dat in de keuzelijst geselecteerd werd. Dat kan door het veld “value” van het select element op te vragen.

Let op: de naam van de afbeelding bevat niet enkel het item uit de keuzelijst, maar ook de taal (`en_` of `fr_`). Werk hier niet met een `if/else`-structuur, maar zorg dat de code makkelijk uitbreidbaar is naar andere talen. (Voor de aandachtige lezers: er zijn enkele afbeeldingen in het Nederlands (`nl`) en Duits (`de`).)

6 Verdere oefeningen op JavaScript

Als je de opgave tot hier afgewerkt hebt, dan heb je de belangrijkste stukken theorie ingeoeft uit de les “*JavaScript in webpagina’s*”. Maar de theorie uit de les “*JavaScript*” kreeg nog niet

veel aandacht. Daar brengen we nu verandering in. Haal er de cursusslides en eventueel een cheatsheet bij, zoals htmlcheatsheet.com/js/.

Haal van Ufora het startproject *webshop* af. Unzip, en open in IntelliJ. De website die je hier vindt, bestaat uit een homepagina en een invulformulier: een redelijk primitieve webshop. Bedoeling is dat deze pagina de rekening van de gebruiker maakt, en desgevallend nagaat of de betaal-/adresgegevens correct ingevuld werden.

We starten alvast met een kleine vingeroefening, en gaan in de volgende paragrafen verder met het schrijven van klassen en rekenwerk op tekst en/of arrays.

In de webshop kan de gebruiker aanvinken welke producten hij wil bestellen (telkens 1 stuk per keer). Als instapoefening schrijf je de JavaScript-code die de totale som van de aangevinkte goederen berekent. Deze totale som toon je in het tekstgebied rechts wanneer op de knop “Voeg toe aan winkelkar” wordt gedrukt. Je mag de oude inhoud van het winkelwagentje ook telkens overschrijven (dus nog geen accumulatie).

Een tip om niet te hard te moeten zoeken in het html-document: gebruik in de browser *inspect* op het tekstgebied om snel de bijhorende html-code te achterhalen.

7 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. Elke klasse komt in een apart bestand.

- 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 in tekstvorm terug te geven (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 in tekstvorm teruggeeft, inclusief totaalprijs.

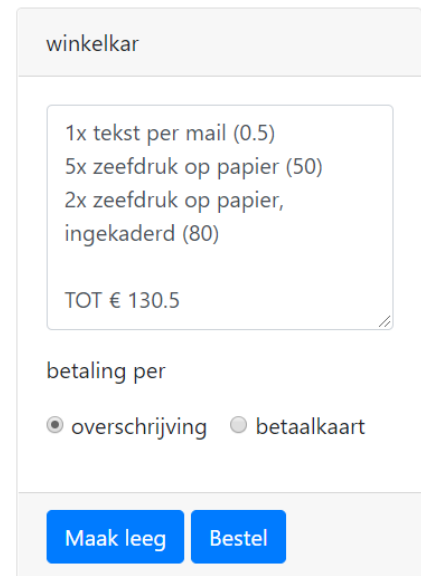
Voorzie een methode die de winkelkar weer leeg maakt.

8 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.

Controleer grondig: kan je een winkelkar vullen, leegmaken en weer vullen? Wat als je producten herhaaldelijk aan de winkelkar toevoegt - verhogen de aantallen zoals het hoort?

Als de website geen reactie geeft, dan is het hoog tijd om te *'inspecteren'* wat er aan de hand is. Eigenlijk moet dit venster constant open staan! (Zie bij *Javascript debuggen*, blz 6.)



winkelkar	
1x tekst per mail (0.5)	
5x zeefdruk op papier (50)	
2x zeefdruk op papier, ingekaderd (80)	
TOT € 130.5	
betaling per	
<input checked="" type="radio"/> overschrijving	<input type="radio"/> betaalkaart
<button>Maak leeg</button>	<button>Bestel</button>

9 Controle vóór het insturen van een formulier

In het html-bestand staat er al code die de bestelling doorstuurt, zodra er op de knop **Bestel** geklikt wordt. 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 verklapt 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. Zoek ook al eens uit hoe je de submit-actie kan annuleren.

10 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. Gebruik de zoektermen `javascript popup for user input`. Of zoek op www.w3schools.com/js/.

localhost:63342 says

Geef de code van uw betaalkaart in

XXXX XXXX XXXX XXXX

OK Cancel

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.

1. Schrijf al zeker een (hulp)functie `cijfersom` die de cijfersom van een getal bepaalt. Laat dit werken voor alle positieve gehele getallen ongeacht uit hoeveel cijfers ze bestaan.

Test uit: print de uitkomst van een aantal functie-oproepen in de console, zoals in onderstaande screenshots getoond. Merk wel op dat daar de *repetitieve* cijfersom berekend werd (zolang de cijfersom groter is dan 9, wordt opnieuw de cijfersom genomen). De niet-repetitieve cijfersom volstaat voor het algoritme van Luhn (leg uit waarom).

Merk op

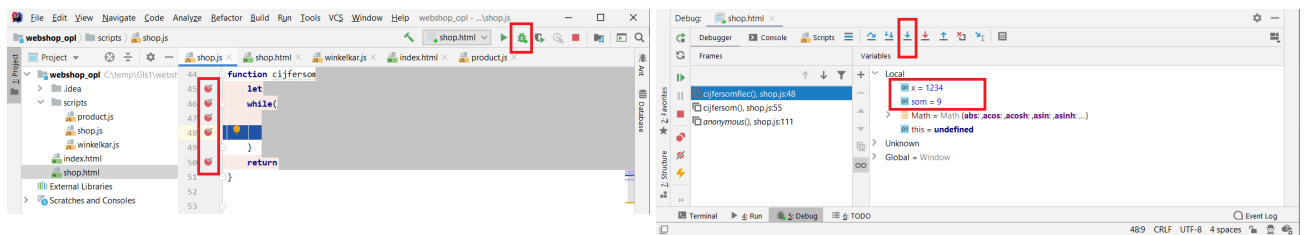
```

console.log(`cijfersom van 12345 is ${cijfersom(x: 12345)} (verwacht: 6)`);
console.log(`cijfersom van 123456 is ${cijfersom(x: 123456)} (verwacht: 3)`);
console.log(`cijfersom van 1234567 is ${cijfersom(x: 1234567)} (verwacht: 1)`);
console.log(`cijfersom van 12345678 is ${cijfersom(x: 12345678)} (verwacht: 9)`);
console.log(`cijfersom van 123456789 is ${cijfersom(x: 123456789)} (verwacht: 9)`);

```

Console		What's New
<div> <div></div> <div></div> <div></div> </div> <div>top</div> <div>Filter</div> <div>Di</div> <div></div>		
cijfersom van 12345 is 6 (verwacht: 6)		shop.js:111
cijfersom van 123456 is 3 (verwacht: 3)		shop.js:112
cijfersom van 1234567 is 1 (verwacht: 1)		shop.js:113
cijfersom van 12345678 is 9 (verwacht: 9)		shop.js:114
cijfersom van 123456789 is 9 (verwacht: 9)		shop.js:115

En zet genoeg breakpoints in de code om te debuggen. In de screenshots hieronder zijn de breakpoints, de debug-icoontjes en de output aangegeven in rode kaders.



2. 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.
3. 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.

11 Wijzigen van 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

☐ volledige tekst van 'Lorem Ipsum' toegestuurd per mail (€ 0.50)

☐ volledige tekst, zeefdruk op handgeschtept papier (€ 10.00)

☒ volledige tekst, zeefdruk op handgeschtept papier, ingekaderd (€ 40.00);

kleur kader

wit

uw gegevens

naam

postcode

straat en nr

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?
