JAVASCRIPT TRAINING

Case 3 - Structurering bestanden

Source

Download de source voor deze opdracht om te beginnen met de case. Pak de zip file uit zoals je in case 1 hebt geleerd. Als het goed is zie je nu in WebStorm onder de folder "opdrachten" de sources van case 3 staan.

Omschrijving

Zojuist is verteld hoe je je javascript kunt opsplitsen in meerdere bestanden. Doel van deze opdracht is herkennen welke stukken javascript logischerwijs in een eigen bestand horen om zo de code overzichtelijk te houden. Daarnaast leer je over de volgordelijkheid van de in te laden code.

Voer voor het uitvoeren van de opdracht het de commando's "npm install" en "grunt watch" uit. Zorg ervoor dat er gedurende de opdracht geen fouten van grunt op de console verschijnen. Je mag ook, i.p.v. het commando "grunt watch" na het uitvoeren van de opdrachten het "grunt" commando draaien, ook hierbij geldt, zorg ervoor dat grunt geen fouten aangeeft.

Hint: de jshint opmerking "The object literal notation {} is preferable." zal pas in de volgende case worden opgelost.

Opdracht 1 - Structurering

Open in je browser de URL "http://localhost:8000/opdrachten/case3/template.html" om de derde case te laden. Je ziet het todoscherm van de applicatie voor je.

Open in WebStorm het bestand "opdrachten/case3/template.html". Bovenin staat op dit moment 1 javascript bestand dat wordt ingeladen. Open dit javascript bestand en probeer te begrijpen wat er gebeurd. Beantwoord onderstaande vragen:

1. Wat is de functie van de TodoView?
2. Wat is de functie van de eventlistener die op het window wordt geplaatst?
3. In hoeveel delen zou je dit bestand opsplitsen?
4. Waarom is de volgorde van de delen die je onderkent hebt in vraag 3 van belang?

Als je bovenstaande vragen hebt beantwoord, probeer dan het bestaande javascript bestand op te splitsen in het aantal delen dat je hebt onderkent in vraag 3. Dit betekend dat je dus net zo veel javascript bestanden krijgt als het aantal onderkende delen. Controleer tevens of de applicatie nog werkt, door in de browser een refresh te geven.

Opdracht 2 - Betere manier van inladen

Als je opdracht 1 afhebt, verwijder dan de eventlistener die op het window wordt geplaatst. Zorg er wel voor dat de code die afgetrapt wordt door de eventlistener blijft staan. Dus het renderen van todos en gebruikers moet wel nog plaatsvinden, alleen niet meer in de eventlistener. Beantwoord onderstaande vragen:

1. Refresh vervolgens je browser, wat gebeurd er nu?
2. Waarom denk je dat de pagina nu niet meer werkt zoals die werkte met de eventlistener?
3. Hoe zorg je ervoor dat de pagina weer werkt als voorheen, zonder de eventlistener terug te plaatsen? Probeer dit uit, en refresh de pagina om te verifieren of alles weer werkt als voorheen.
4. Welke voordelen biedt de oplossing die je in vraag 3 hebt bedacht nog meer?

Opdracht 3 - Extra opdracht

Mocht je tijd overhebben na het maken van bovenstaande opdrachten, dan kun je je middels deze opdrachten verdiepen in het inladen van javascript middels javascript. Op dit moment heb je in het template een aantal javascript tags staan waarmee je de verschillende bestanden inlaad.

Probeer middels deze opdracht ervoor te zorgen dat er maar 1 javascript bestand wordt geladen vanuit het template. Dit script bevat dan de initialisatie code van de applicatie, en middels dit initialisatie script dien je in deze extra opdracht de andere javascript bestanden in te laden. Tijdens het uitvoeren van deze opdracht komen enkele concepten voor die nog niet zijn behandeld. Daarom hieronder wat tips.

Tips:

- Middels `var nieuweScriptTag = document.createElement("script");` kun je programmatisch een scripttag maken.

- Middels `nieuweScriptTag.setAttribute("key", "value");` kun je attributen zetten.
- Gebruik de case 2 geleerde selectors om de scripttag aan de body toe te voegen.
- Gebruik `nieuweScriptTag.onload = function() {}` om code uit te voeren als een script klaar is met laden. Dit mechanisme heet een callback.

PUBLISHED ON June 30th, 2015