JAVASCRIPT TRAINING

Case 5 - Inheritance & Scoping

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 5 staan.
Presentatie in PDF

Omschrijving

Zojuist is verteld hoe je inheritance via prototyping werkt in javascript en in welke scope objecten zitten die je aanmaakt. Tevens is je net uitgelegd hoe het module pattern werkt. Het doel van deze opdracht is het leren van omgaan met inheritence van objecten middels prototyping. Tevens leer je in deze opdracht hoe je ervoor zorgt dat niet alle objecten die je maakt in de global scope terecht komen.

Opdracht 1 - Inheritance via prototyping

Open in je browser de URL "http://localhost:8000/opdrachten/case5/template.html" om de zesde case te laden. Je ziet het todoscherm van de applicatie voor je.

Bekijk in WebStorm de javascript bestanden die bij deze opdracht horen. Doorloop onderstaande stappen om de opdracht te voltooien.

Inheritance

1. De bedoeling is dat we aan de TodoView class standaard logfunctionaliteiten gaan gebruiken die in de View class is gedeclareerd. Zorg ervoor dat de TodoView class de View class extend.
2. Refresh in je browser de pagina en gebruik FireBug om de todoViewInstance die in init.js gemaakt wordt op de console te loggen. Als het goed is geeft FireBug aan dat de todoViewInstance een constructor heeft "TodoView" en 3 methodes; renderTodos, renderGebruikers en logVariable. Als je Chrome gebruikt, dan zie je als het goed is dat TodoView extend van View, doordat het __proto__ attribuut naar View verwijst, waar de logVariable is gedeclareerd.
3. Voer in FireBug/Chrome een instanceof check uit van de todoViewInstance tegen View en tegen TodoView. Als het goed is leveren beide resultaten true op. Als dit niet zo is, heb je de prototype chain in stap 1 niet goed opgezet.
4. Zorg in de renderTodos en renderGebruikers methodes ervoor dat als eerste regel in die methodes de logVariable methode wordt aangeroepen op de this instance. Geef aan de methode de gebruikers/todos waarmee dus de inputvariabele van de methode worden gelogd.
5. Refresh in je browser de pagina. Controleer of er twee logregels op de FireBug console verschijnen. Als het goed is zie je twee logregels met 1 maal de gebruikers array als JSON en 1 maal de todos array als JSON gelogged. De prefix is echter nu nog "undefined", hoe kan dit? Waar komt deze prefix vandaan? Los dit probleem op door een aanpassing te maken in de TodoView class, gebruik als prefix "TodoView".
6. Refresh nogmaals de browser, als je stap 5 goed hebt gedaan zie je nu "TodoView" als prefix voor de logregel.

Overriden methods

7. Alle logregels die vanuit de TodoView gelogd worden, dienen nu tevens de lengte van de Array te loggen. Om dit te bewerkstelligen moet je de logVariable methode overriden in de TodoView. Doe dit, en zorg er tevens voor dat de logVariable methode uit de View class (de parent of superclass) wordt aangeroepen. Voer daarna het tweede log statement uit in de overridden methode. Het aantal element van een Array kun je opvragen via de "length" property. Als je de browser refreshed dien je in de console log van je browser het volgende te zien:

```
TodoView - {"id":1,"titel":"Afmaken opdracht 1",...
Aantal items gerenderd: 2
TodoView - {"id":1,"naam":"J. Script","gebruikersnaam":"jscript"...
Aantal items gerenderd: 3
```

Opdracht 2 - Scoping

Als je de javascript bestanden van case 5 bekijkt, dan zie je dat er een aantal zaken worden gedeclareerd, zoals: todos, gebruikers, todoViewInstance, maar ook classes als; TodoView en View. Je hebt je misschien al afgevraagd wat er gebeurd als je nogmaals variabelen declareert als "todos" of "gebruikers". Tevens heb je je misschien afgevraagd wat er gebeurd als je nog een "TodoView" declareert.

Als je geluk hebt bij het dubbel declareren van instance variabelen en classes krijg je een TypeError in je console, als je pech hebt lijkt alles nog te werken zoals het hoort, totdat je tegen een uithoek van je applicatie aanloopt en je vreemd gedrag en javascript errors krijgt. Volg onderstaande stappen om de opdracht uit te voeren:

1. Probeer in init.js bovenaan nogmaals een constructor function genaamd TodoView te declareren. Refresh vervolgens je browser, wat gebeurd er? Hoe komt dit?
2. Haal de net gemaakt constructor function weer weg, refresh je browser en verifieer of de pagina weer werkt. Voer middels FireBug een console.log uit van de variable "todoViewInstance" en van "window.todoViewInstance". Wat valt je op? Welke conclusie kun je hieruit voorzichtig trekken? Voer een === vergelijking tussen de twee uit om dit te verifieren.
3. Herhaal stap 2 met de variabele "gebruikers". Herhaal stap 2 vervolgens met de classes "View" en "TodoView". Wat valt je op?
4. Op welke manier kunnen we dit probleem van scoping oplossen, dus hoe zorg je ervoor dat niet alles binnen de global scope zit?
5. Pas het module pattern toe en zorg ervoor dat je een DataModule en een ViewModule maakt. Zorg er voor dat de applicatie na de aanpasingen blijft werken door een refresh te geven in je browser en te controleren dat er geen javascript errors meer optreden.

Tips bij stap 5:

- Voor deze opdracht is het voldoende om alle code die in 1 module hoort, ook in 1 file/bestand te plaatsen. Vergeet het template hier niet op aan te passen.
- Denk goed na over de publieke API van de modules. Niet alle interne code hoeft van buiten de module benaderbaar te zijn.
- Als je de opdracht goed hebt uitgevoerd, zijn alleen module variabelen en de publieke API's van de modules die hebt gemaakt beschikbaar in de global scope. Zo zal de View class, TodoView class en de todoViewInstance in de init.js niet meer beschikbaar moeten zijn in de global scope. Toon dit aan/controleer door stap 2 te herhalen. De View class zou je helemaal niet meer moeten kunnen achterhalen middels FireBug.

PUBLISHED ON February 13th, 2015