Overzichtsoefening 1

Course.js

Maak de Javascript klasse Course in het bestand src/school/Course.js.

Course heeft eigenschappen _id, _grade en _completed.

De constructor heeft als argument een waarde voor de eigenschap _id. _grade wordt gelijk gesteld aan 0 en _completed wordt gelijk gesteld aan false.

Voorzie getters en setter voor alle eigenschappen.

Je mag er vanuit gaan dat _grade altijd een geheel getal bevat tussen 0 en 20.

Student.js

Maak de Javascript klasse Student in het bestand src/school/Student.js en

Student heeft eigenschappen _id en _courses.

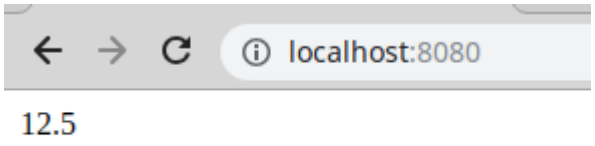
De constructor heeft als argument een waarde voor de eigenschap id. _courses wordt gelijk gesteld aan een lege array.

Voorzie de function addCourse. Deze functie verwacht een Course-object als argument. Als het argument geen Course is wordt een Error opgeworpen (je kan dit controleren via instanceof). Indien het argument wel een Course is, wordt de course toegevoegd aan de array _courses (een element toevoegen aan een array kan via push).

Maak de function calculateGrade. In deze functie worden alle elementen uit _courses doorlopen. Per completed course wordt de waarde van de grade opgeteld bij de som. De totale eindscore (grade) die teruggekeerd wordt, is de som gedeeld door het aantal cursusen die completed zijn. Wanneer het aantal completed courses echter gelijk is aan 0, wordt een error opgeworpen.

Via index.html en app.js kan je de toepassing testen.

De output is van de vorm:



Student.test.js

Voorzie 2 (Jest) tests in het bestand test/js/school/Student.test.js.

(1) De functie calculateGrade moet een error opwerpen wanneer hij aangeroepen wordt voor een student-object die nog geen completed course heeft.

(2) De functie calculateGrade moet 12 als terugkeerwaarde hebben voor een student die één (completed) course heeft met grade 12.

Overzichtsoefening 2

```
src
└── js
    ├── app.js
    └── cargo
        ├── Container.js
        └── Product.js
```

Maak in de map oef1/src/js/cargo/Product.js de klasse Product.

Product heeft heeft eigenschappen _id en _weight (gewicht)

De constructor heeft als argument een waarde voor id en weight. Deze waarden worden toegekend als

- id een geheel getal is groter dan 0
- weight een getal is groter dan 0

Indien de waarden niet voldoen dan wordt een Error opgeworpen.

Hint: Number.isInteger(id)
typeof weight != 'number'

Verder worden ook getters voor alle eigenschappen aangeboden in de klasse Product.

Maak in de map oef1/src/js/cargo/Container.js de klasse Container.

Binnen een Container kunnen producten bijgehouden worden in de array _products . Ook wordt het maximale gewicht dat de container kan bevatten bijgehouden in het veld _maxWeight.

In de constructor wordt een waarde als argument meegegeven voor maxWeight. maxWeight moet een number zijn groter dan 0. Indien de waarde voor maxWeight niet voldoet wordt een Error opgeworpen. Ook wordt in de constructor _products gelijk gesteld aan een lege array.

Hint: typeof maximumWeight != 'number'

Via de methode addProduct wordt een product toegevoegd aan de array _products. Het product, dat als argument meegegeven wordt, moet voldoen aan een aantal voorwaarden:

- product moet een object zijn van de klasse Product
- het id van product mag nog niet gebruikt zijn als id van een van de producten in _products
- de som van alle gewichten in _products en het gewicht van het nieuwe product mag niet groter zijn dan _maxWeight

Indien niet aan een van de bovenstaande voorwaarden voldaan is wordt een Error opgeworpen.

Hint: !(product instanceof Product)

Via de methode getProductAtIndex wordt het product teruggeven dat op de als argument meegegeven index in de array_products staat.
Het argument index van deze methode moet voldoen aan de volgende voorwaarden:

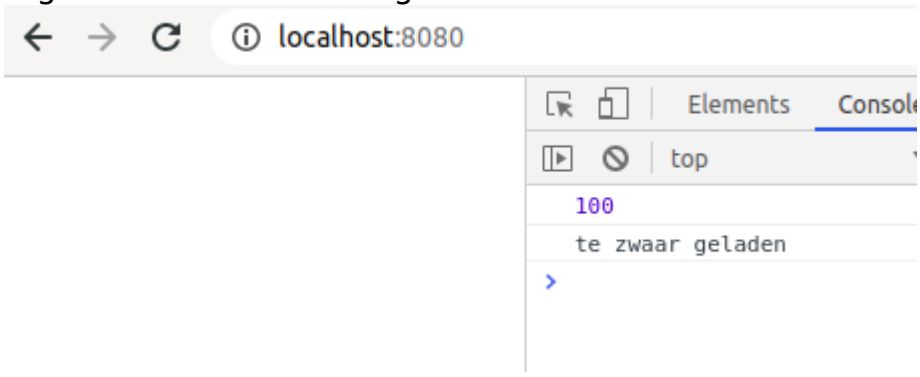
- index moet een geheel getal zijn
- index moet groter dan 0 zijn
- index moet kleiner zijn dan de lengte van de array _products

Indien niet aan de bovenstaande voorwaarden voldaan is wordt een Error opgeworpen.

De werking van deze klassen wordt geïllustreerd aan de hand van onderstaande code:

```
"use strict";
import Container from './cargo/Container';
import Product from './cargo/Product';
try{
    let product1 = new Product( 1,200 );
    let product2 = new Product( 2,100 );
    let product3 = new Product( 3,400 );
    let container = new Container( 700 );
    container.addProduct( product1 );
    container.addProduct( product2 );
    console.log( container.getProductAtIndex(1).weight);
    container.addProduct( product3 );
} catch(error){
    console.log(error.message);
}
```

Afgedrukt in de console geeft dit onderstaand resultaat:



Maak unit-tests in de map tests waarin het volgende gecontroleerd wordt:

addProduct

- er wordt een Error opgeworpen wanneer addProduct aangeroepen wordt met de string “test” als argument
- er wordt een Error opgeworpen wanneer addProduct aangeroepen wordt met een product als argument waarvan het id al voorkomt als id van één van de producten in _products
- er wordt een Error opgeworpen wanneer addProduct aangeroepen wordt met een product waarvoor geldt dat de som van alle alle gewichten in _products en het gewicht van het nieuwe product groter is dan maxWeight

Overzichtsoefening 3

Installeer JSHint (of een alternatief https://blog.logrocket.com/four-options-to-help-you-get-started-linting-your-javascript-b4b829828648/) en probeer uit met de code van de oefeningen en opdrachten.
De installatie gebeurt via het volgende commando

npm install jshint --global

(of
sudo npm instlal jshint --global
)

in de map van je project plaats je het bestand .jshintrc met als inhoud

{
"esversion": 6
}

Vervolgens gebruik je het commando
jshint ...
met op de drie puntjes de lokatie van je bestand.