AngularJS: modules en controllers

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

In dit hoofdstuk

- Wat is een module ?
- Wat is een controller ?
- Wat is $scope ?
- Functies schrijven in de controller en de directive ng-click
- Module en controller in aparte bestanden
- Andere syntaxis om controllers te definiëren
- De controllerAS-syntaxis

Wat is een module ?

- Inleiding
  - Het is belangrijk in een AngularJS app modules en controllers te definiëren omwille van de herbruikbaarheid van alle logica.
  - De app is op die manier goed gestructureerd in zowel eigen, afzonderlijke functiegebieden alsook voor bepaalde stukjes user interface

Wat is een module ?

- Inleiding
  - In vrijwel elke Angular-app is het gebruikelijk om een modulenaam te definiëren in de directive ng-app
  - Angular gaat dan op zoek naar de bijhorende JavaScript-module waarin de logica voor de app is gedefinieerd

```
<html ng-app="myApp">
```

Wat is een module ?

- Inleiding
  - creatie modulenaam (1ste manier)
    - In JavaScript wordt een module als volgt gemaakt:

```
<script>
    var app = angular.module('myApp', []);
</script>
```

    - Een globale variabele app bevat alle logica voor AngularJS-modules
    - Globale variabelen worden liefst zoveel mogelijk vermeden

Wat is een module ?

- Inleiding
  - creatie modulenaam (2de manier)
    - Zonder JavaScript-variabelen:

```
<script>
    angular.module('myApp', []);
</script>
```

Wat is een module ?

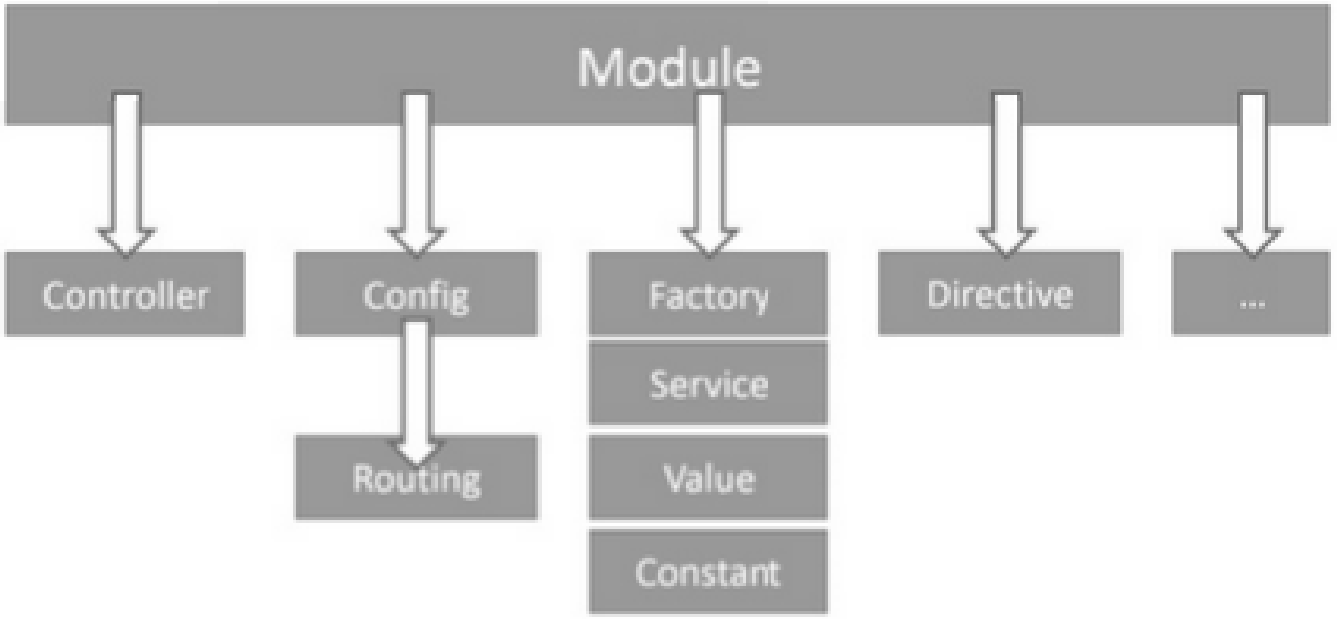
- Inleiding
  - Een voorbeeld

```
<html ng-app="myApp">
<head>
    ...
</head>
<body>
    <div class="container" >
        <h1>Een eenvoudige module myApp</h1>
        <p>Bekijk de variabele 'app' via de Dev Tools console.</p>
    </div>
    <script src="../js/vendor/angular/angular.js"></script>
    <script>
        // er wordt een globale variabele app gemaakt
        var app = angular.module('myApp', []);
    </script>
</body>
</html>
```

Wat is een module ?

- Inleiding
  - Basisarchitectuur van een AngularJS-app
    - Een module is de belangrijkste container
    - Er kan naar keuze functionaliteit aan worden toegevoegd

```
<html ng-app="mcduleName">
```



Wat is een module ?

- Module als setter en getter
  - Een modulenaam, opgegeven met blokhaken, fungeert als een setter. De module wordt door Angular gemaakt en in het geheugen gezet
  - Een modulenaam, opgegeven zonder blokhaken, fungeert als een getter. Angular zoekt in het geheugen naar de module en voert er handelingen mee uit

In code ziet dat er zo uit:

```
angular.module('myApp', []); // angular.module() is setter; module wordt gemaakt
angular.module('myApp'); // angular.module() is getter; module wordt opgezocht
```

Wat is een module ?

- Dependencies (script_0301)
  - Een module met lege blokhaken staat volkomen op zichzelf
  - Een module met blokhaken is een JavaScript-array
  - Deze array wordt gebruikt om extra functionaliteit op te laden
  - De door komma's gescheiden modulenamen worden door Angular via het DI-mechanisme geladen

```
angular.module('myApp', ['ngRoute', 'ngAnimate', '…']);
```

Wat is een controller ?

- Inleiding
  - Volgens goed AppDesign worden structuur en logica van elkaar gescheiden.
  - In een Angular-app wordt de logica (ook wel: business logic of business intelligence) door controllers verzorgd
  - Controllers zijn gewoon JS-functies

Wat is een controller ?

- De directive ng-controller
  - In HTML wordt met deze directive aangegeven welke controller van toepassing is op een bepaald deel

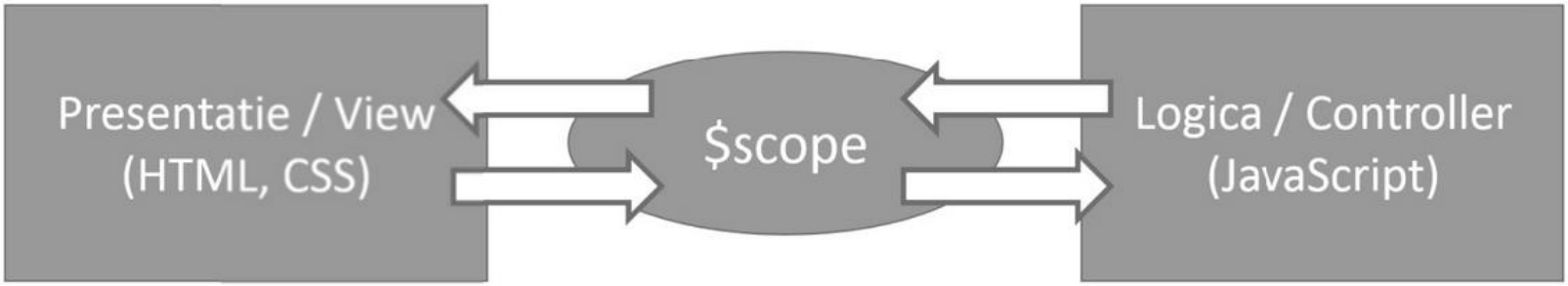
```
<div id="menu" ng-controller="menuController">
    ...
</div>
```

Wat is een controller ?

- Wat is $scope ?
  - $scope is een JavaScript-object dat door AngularJS als transportmechanisme voor variabelen tussen de view en de Controller wordt gebruikt
  - Aan het object $cope kunnen variabelen en functies toegevoegd worden die samenwerken met de HTML van de pagina

Wat is een controller ?

- Wat is $scope ?



De relatie tussen de view en de controller. Het object $scope verbindt ze met elkaar.

Wat is een controller ?

- Een controller definiëren (script_0302)

```
<div class="container" ng-controller="testController">
    <h1>Eenvoudige controller</h1>
    <h2>{{ name }}</h2>
    <p>Favoriete stad: {{ city }}</p>
    <p>Favoriet land: {{ country }}</p>
</div>
```

en de bijhorende controller:

```
var app = angular.module('myApp', []);
app.controller('testController',
    function ($scope) {
        $scope.name = 'Peter Kassenaar';
        $scope.city = 'Groningen';
        $scope.country = 'Verenigde Staten';
    });
```

Wat is een controller ?

- Een controller definiëren (script_0302)

De logica van de controller wordt via $scope doorgegeven en in de view geplaatst.

Wat is een controller ?

- Een uitgebreider voorbeeld (script_0303)
  - Html-code

```
<div class="container" ng-controller="myController">
    <h1>Filteren op achternaam</h1>
    <p>
        <input type="checkbox" ng-model="reversed" />Omgekeerd sorteren
    </p>
    <p>
        <input type="text" ng-model="naam" class="input-lg" placeholder="filter op naam..." />
    </p>
    <ul>
        <li ng-repeat="person in persons | orderBy: 'name' : reversed | filter
            : {lastname: naam } as numResults ">
            {{ person.name }} - {{ person.lastname }}
        </li>
        <li ng-if="numResults.length === 0">
            <strong>Geen resultaten gevonden...</strong>
        </li>
    </ul>
</div>
```

Wat is een controller ?

- Een uitgebreider voorbeeld (script_0303)
  - Controller-code

```
app.controller('myController',
    function ($scope) {
        $scope.persons = [
            { 'name': 'John', lastname: 'Visser' },
            { 'name': 'Bob', lastname: 'Tasseur' },
            { 'name': 'Bart', lastname: 'Bever' },
            // …];
        $scope.reversed = true;
    });
```

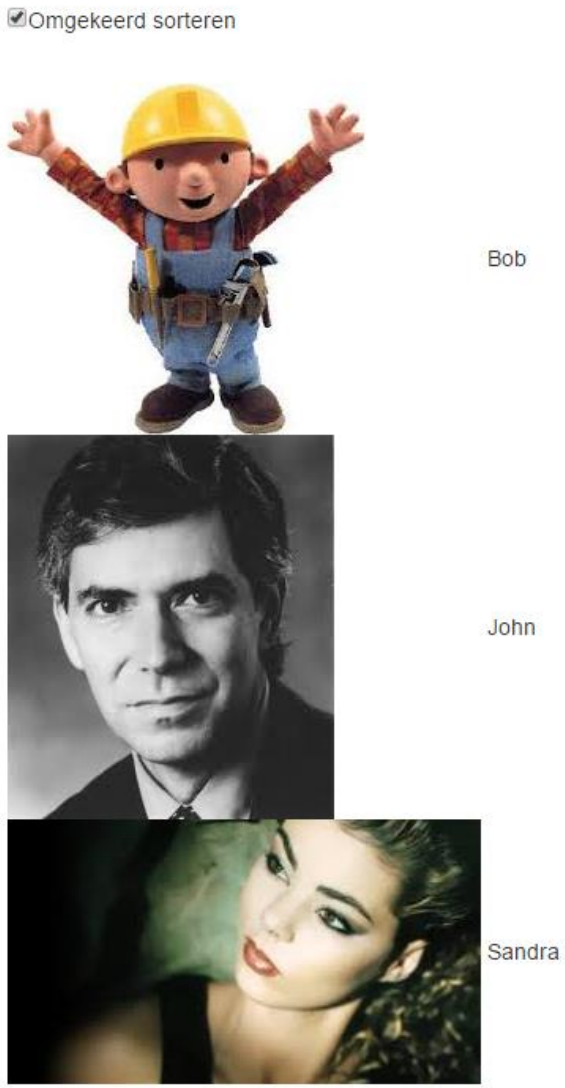
Wat is een controller ?

- Een uitgebreider voorbeeld (script_0303)
  - Resultaat in de browser

De gegevens zijn nu uit de controller afkomstig.

Oefening op controllers

- Neem de oefening uit het vorige hoofdstuk “foto_tonen.html” en refactor deze zodanig dat het geheel werkt onder een controller
- Zorg ervoor dat de checkbox default “enabled” staat
- Geef je project de naam “foto_tonen_controller.html”



Functies in de controller & directive ng-click

- Inleiding
  - Naast enkelvoudige waarden of arrays kunnen ook functies toegevoegd worden in de controller.
  - Deze werken dan in combinatie met event handlers in de view
  - In Angular zijn hiervoor diverse directives beschikbaar (ng-click, ng-mouseover, ng-blur, …)

Functies in de controller & directive ng-click

- Een voorbeeld (script_0304)

```
<button ng-click="clickMe()" class="btn btn-lg btn-primary">Klik op mij!</button>
```

```
app.controller('myController',
    function ($scope) {
        …
        $scope.clickMe = function () {
            alert('Er is op de knop geklikt!');
        };
    });
```

Functies in de controller & directive ng-click

- ng-click binnen ng-repeat
  - In plaats van een ‘losse’ **ng-click** als **event handler** voor een knop te schrijven, mag hij bijvoorbeeld ook worden opgenomen binnen een ng-repeat
  - Een moeilijker voorbeeld (script_0305)
    - Als op een naam uit een HTML-lijst wordt geklikt, worden de details van de betreffende persoon in de pagina getoond

Functies in de controller & directive ng-click

- ng-click binnen ng-repeat
  - Een moeilijker voorbeeld (script_0305)
    - html-code

```
<!-- 1. De lijst met namen -->
<ul>
    <li ng-repeat="person in persons">
        <a href="#" ng-click="showDetail(person)">
            {{ person.name }} {{ person.lastname }}
        </a>
    </li>
</ul>
<hr />
<!-- 2. Details per persoon -->
<div>
    <h3>{{ voornaam }} {{ achternaam }} </h3>
    <strong>{{ leeftijd }} jaar </strong>,
    <a href="mailto:{{ email }}">{{ email }}</a>
</div>
```

Functies in de controller & directive ng-click

- ng-click binnen ng-repeat
  - Een moeilijker voorbeeld (script_0305)
    - controller-code

```
app.controller('myController',
    function ($scope) {
        // 1. array met namen
        $scope.persons = [
            { 'name': 'John', lastname: 'Visser', age: 23, email: 'john@visser.nl' },
            //… overige namen uitgebreid
];

        // 2. klikken op een naam afvangen
        $scope.showDetail = function (person) {
            $scope.voornaam = person.name;
            $scope.achternaam = person.lastname;
            $scope.leeftijd = person.age;
            $scope.email = person.email;
        }

        // 3. initiele waarde instellen
        $scope.showDetail($scope.persons[0]);
    });
```

Functies in de controller & directive ng-click

- ng-click binnen ng-repeat
  - Een moeilijker voorbeeld (script_0305)
    - resultaat in de browser

Functies in de controller & directive ng-click

- Meer kenmerken van controllers
  - Er kunnen meerdere controllers op een pagina gedefinieerd worden. Met ng-controller wordt er aangegeven welke bedoeld wordt
  - Controllers worden telkens opnieuw geïnitialiseerd. Eventuele wijzigingen blijven niet bewaard. Hiervoor moet de ontwikkelaar zelf voorzieningen treffen

Oefeningen

- https://bb.pxl.be
  - Leermaterialen/week1/oefeningen
    - Oef=>H3 – Modules en controllers
      - Oefening 1
      - Oefening 2

Module en controller in aparte bestanden

- Modulaire programma's (script_0306)
  - Vanuit het oogpunt van modulair programmeren is het een goed idee de modules en controllers in aparte bestanden te zetten en in de HTML-pagina's een scriptreferentie naar die bestanden op te nemen

Module en controller in aparte bestanden

- Modulaire programma's (script_0306)
  - HTML
  - Module in app.js
  - Controller in controller.js

Module en controller in aparte bestanden

- Modulaire programma's (script_0306)
  - HTML

```
<html ng-app="myApp">
<head>
    …
</head>
<body>
    <div class="container" ng-controller="myController">
    …
    </div>
    <!-- Vendor scripts -->
    <script src="../js/vendor/angular/angular.js"></script>

    <!-- Custom scripts -->
    <script src="../js/H03/script_0306_app.js"></script>
    <script src="../js/H03/script_0306_controller.js"></script>
</body>
</html>
```

Module en controller in aparte bestanden

- Modulaire programma's (script_0306)
  - Module in app.js

```
// app.js - maak de hoofd-module voor de app
(function () {
    angular.module('myApp', []);
})();
```

Module en controller in aparte bestanden

- Modulaire programma's (script_0306)
  - Controller in controller.js

```
// controller.js
(function () {
    angular.module('myApp') // angular.module() fungeert nu als getter.
        .controller('myController', function ($scope) {
            // …

        });
})();
```

Andere syntaxis om controllers te definiëren

- Minification safe syntaxis
  - Bij het maken van een controller wordt na de naam van de controller als tweede parameter een array meegegeven
  - In de array staan vervolgens op volgorde de parameters zoals de controller ze verwacht
  - Het laatste element in de array is tenslotte en altijd de controllerfunctie zelf

```
angular.module('myApp')
    .controller('myController', ['$scope', function($scope){
        // …implementatie
}]);
```

Andere syntaxis om controllers te definiëren

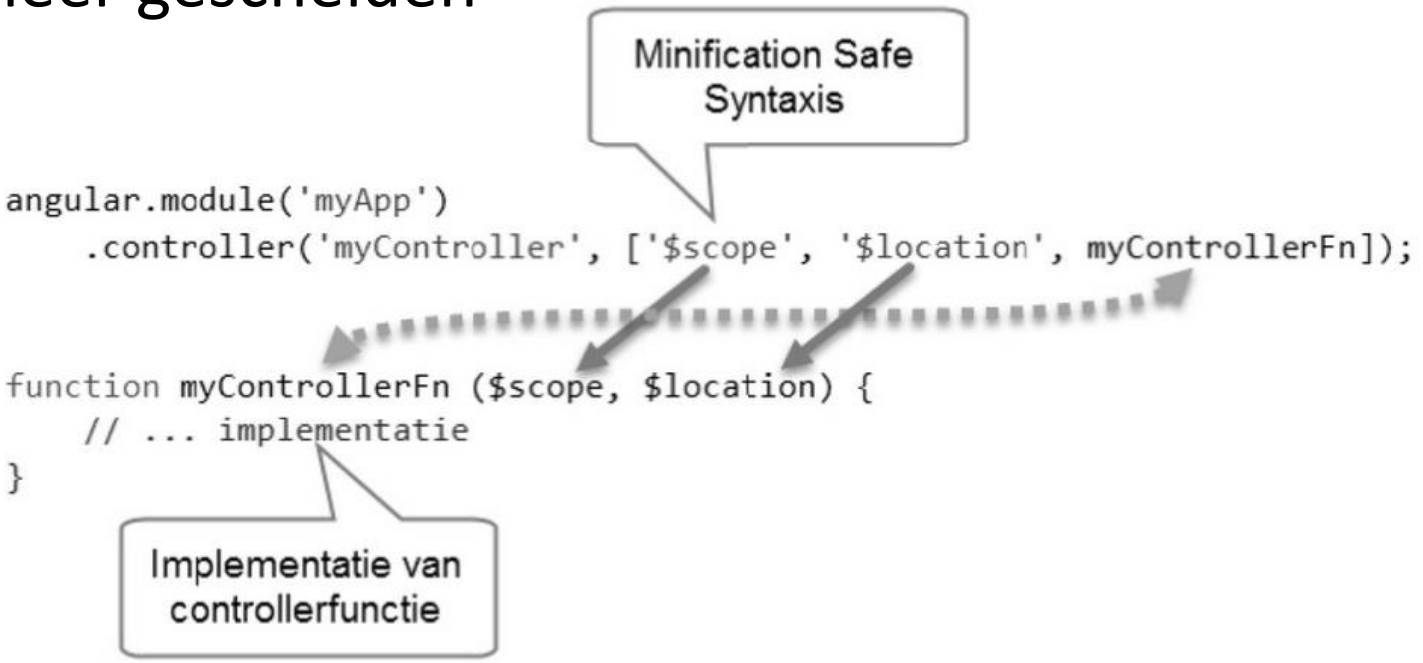
- Benoemde functies
  - Controller wordt niet als anonieme, inline functie gedefinieerd maar krijgt een benoemde functie
  - In minification safe syntaxis wordt dat:

```
angular.module('myApp')
    .controller('myController', ['$scope', '$location', myControllerFn]);

    function myControllerFn($scope, $location){
        // …implementatie
});
```

Andere syntaxis om controllers te definiëren

- Benoemde functies
  - Definitie en implementatie van de controller zijn meer gescheiden



Andere syntaxis om controllers te definiëren

- Werken met de service $inject
  - Afhankelijkheden kunnen apart geïnjecteerd worden
  - Definitie en implementatie van de functie zijn daardoor gescheiden
  - Deze techniek wint aan populariteit en wordt aangeraden te gebruiken

```
// 1. Definitie
angular.module('myApp')
    .controller('myController', myControllerFn);
    // 2. Dependencies injecteren
    myControllerFn.$inject = ['$scope', '$location'];

    // 3. Implementatie van myControllerFn
function myControllerFn($scope, $location){
        // …
});
```

De controllerAs-syntaxis

- Inleiding
  - Een andere manier om te werken met de controller
  - Heeft aan populariteit gewonnen
  - De toekomstige manier om sinds AngularJS 1.3 en hoger controllers te maken

De controllerAs-syntaxis

- Weg met $scope
  - Geen $scope-variabele meer om gegevens uit te wisselen tussen controller en view
  - Alle variabelen worden rechtstreeks op de controller geplaatst (via het het JavaScript-keyword this)
  - Vaak wordt een alias gebruikt voor het keyword (voorbeeld: var vm =this)

De controllerAs-syntaxis

- Var vm als viewmodel
  - Een voorbeeld

In script_0307 is de controllerAs-syntaxis gebruikt. Voor het overzicht is alle code in één bestand geplaatst. Zelf weet u dat ze ook zouden kunnen worden gesplitst over meerdere HTML- en JavaScript-bestanden.

```
<div class="container" ng-controller="myController as ctrl">
    …
    <ul>
        <li ng-repeat="person in ctrl.persons">
            <a href="#" ng-click="ctrl.showDetail(person)">
                {{ person.name }} {{ person.lastname }}
            </a>
        </li>
    </ul>
    <hr />
    <div>
        <h3>{{ ctrl.voornaam }} {{ ctrl.achternaam }} </h3>
        <strong>{{ ctrl.leeftijd }} jaar </strong>,
        <a href="mailto:{{ ctrl.email}}">{{ ctrl.email }}</a>
    </div>
</div>
```

De controllerAs-syntaxis

- Var vm als viewmodel
  - Een voorbeeld (vervolg)

En de controller is als volgt. Let op het ontbreker van $scope:

```
angular.module('myApp', []) // module maken, geen dependencies
    .controller('myController', myController);

function myController() {
    // 1. cache de huidige controller ('this') in
    // een variabele vm (=afkorting voor ViewModel)
    var vm = this;

    // plaats eigenschappen en functies rechtstreeks op de controller
    vm.persons = [
        { 'name': 'John', lastname: 'Visser', … },
        …
        ];

    // 2. klikken op een naam afvangen
    vm.showDetail = function (person) {
        // 2a. Gelukkig is de controller gecached in 'vm',
        // want 'this' heeft binnen deze functie een andere scope!
        vm.voornaam = person.name;
        vm.achternaam = person.lastname;
        vm.leeftijd = person.age;
        vm.email = person.email;
    }

    // 3. initiele waarde instellen
    vm.showDetail(vm.persons[0]);
};
```

De controllerAs-syntaxis

- Conclusie

De uitvoer is opnieuw dezelfde als in de vorige paragrafen. Maar nu is hij tot stand gekomen op basis van AngularJS *best practices*. Nogmaals, onze verwachting is dat de controllerAs-syntaxis sterk in belang zal toenemen en ook in Angular 2.0 een grote rol gaat spelen.

AngularJS Style Guide

Als u op zoek bent naar *best practices* omtrent de notatiewijze en implementatie van modules, controllers, directives en meer, lees dan ook de AngularJS Style Guide op Github. Deze staat onder beheer van John Papa, een GDE (*Google Developer Expert*) op het gebied van AngularJS. Het adres is **github.com/johnpapa/angularjs-styleguide**.

Oefeningen

- https://bb.pxl.be
  - Leermaterialen/week1/oefeningen
    - Oef=>H3 – Modules en controllers
      - Oefening 3
      - Oefening 4