OEFENINGEN ANGULARJS

01 – EENVOUDIG BEGINNEN

a) Maak een eenvoudige AngularJS-applicatie met data binding.
   - Begin een nieuw project.
   - Voeg AngularJS in via een CDN of download een lokale kopie.
   - Maak een pagina waarop je met `{{ … }}` een data binding expressie evalueert (zoals `2 + 2`, of complexer JavaScript).
   - Breid de pagina uit met een tekstvak en de directive `ng-model`. Toon de inhoud van het tekstvak live in de pagina.

b) Bekijk de `demo 02_ng_repeat.html`. Maak daarna zelf een lijst op de pagina waarin de inhoud van een data-model in een lus wordt getoond. Gebruik hiervoor de directive `ng-repeat`.

c) Breid de lijst uit met een filter. Toon bijvoorbeeld alleen de items in de lijst waarvan de tekst in een tekstvak staat. Zie eventueel als demo `03_ng_filter.html`.
   - Breid je demo uit met enkele aanvullende, ingebouwde filters zoals `uppercase`, `lowercase` of `orderby`.
   - Maak een filter waarbij een lijst in omgekeerde volgorde wordt gesorteerd (`reversed`)
   - Zie voor documentatie http://docs.angularjs.org/api/ng/filter.

02 – MODULES EN CONTROLLERS

a) Geef je `ng-app` directive een waarde, dit wordt de naam van de module (bijvoorbeeld `myApp`). Maak vervolgens in JavaScript de module.
   - Wat gebeurt er als je de blokhaken `[]` weglaat in de moduledefinitie en de pagina opent?

b) Wijzig je demo, zodanig dat de data nu door een controller op de hiervoor gemaakte module wordt geleverd. De HTML-code (en werking) blijft dan gelijk, maar op de achtergrond komt het resultaat op andere wijze tot stand.
   - Als voorbeelden zijn eventueel `06_ng_controller01.html` en verder beschikbaar (maar probeer het ook zelf)
   - De controller staat (voorlopig) in de global scope. Test dit via de Chrome Developer Tools Console. Later gaan we dit met de iffy-notatie oplossen.

c) Breid de `$scope` voor de controller uit met extra variabelen. Denk bijvoorbeeld aan NAW-velden, klanteigenschappen, productbeschrijvingen en meer. Toon de `$scope`-variabelen in de HTML-pagina.

d) Maak je demo geschikt voor two-way data binding:
   - zorg er voor dat de user gegevens in een tekstvak kan typen.
   - Sla deze gegevens op in een `$scope`-variabele en toon de tekst in de pagina.
   - Denk bijvoorbeeld aan een tekstbox waarin iemand zijn favoriete stad of land kan typen. Toon deze in de pagina, en bewaar hem in de `$scope`.

e) Voeg een tweede controller toe aan je module. Denk aan de directive `ng-controller` in de HTML en `angular.module('..').controller('naamController', controllerFunctie)` in je JavaScript. Test of:
   - …je deze controller side-by-side met de andere controller kunt gebruiken.
   - …je deze controller kunt nesten *binnen* de andere controller.
   - Zet op beide controllers een `$scope`-variabele met *dezelfde* naam. Welke waarde is in welke controller zichtbaar je HTML?

03 – REFACTORING

a) Refactor je controller(s) zodanig dat deze een minification-safe syntaxis gebruikt. Gebruik beide opties:
   - De array-notatie `['$scope', function ($scope){ …}]);`
   - De `$inject`-notatie, bijvoorbeeld `personController.$inject =['$scope'];`
   - Zie eventueel `06_ng_controller_minifySafe01.html` en `-02` als voorbeeld.

b) Wijzig je applicatie met controller of maak een nieuwe controller. Gebruik nu de controllerAs-syntaxis.
   - Gebruik het keyword `this` voor variabelen in de controller.
   - Wijzig de HTML zodanig dat een `controllerAs`-alias wordt gebruikt.
   - Denk aan het alias-voorvoegsel in de data binding (als in `vm.firstName` en meer) !

c) Dependency Injection: maak een nieuwe module en injecteer deze in je eerdere module.
   - Definieer op de nieuwe module bijvoorbeeld een controller die je in de hoofdmodule aanroept.
   - Zie eventueel `07_ng_module_DI.html` als voorbeeld.

d) Refactor de code zodanig dat je nu gescheiden bestanden hebt. De HTML-code in een HTML-pagina, en de JavaScript/AngularJS-code in aparte JavaScript-bestanden. Deze worden ingevoegd in de hoofdpagina.
   - Zie voor een demo `08_ng_module_refactor.html`.
   - Gebruikelijk is om je module te bootstrappen in een bestand `app.js`.
   - De controllers en andere componenten worden daarna gedefinieerd/gereferenced.
   - Gebruik iffy-notatie in app.js en de andere bestanden.

04 – ROUTING

a) Maak een routing-table voor je app en toon verschillende views in de hoofdpagina. Hanteer deze volgorde:
   - Download en reference eerst de routing-module in de HTML-pagina. Dit is `angular-route.js`.
   - Maak een `<div>` waarin de views worden getoond. Deze krijgt de directive `ng-view`.
   - Plaats je HTML-code die nu nog in de hoofdpagina stond in diverse .html-bestanden in een map `/partials` of `/views`.
   - Injecteer `ngRoute` in de AngularJS-module en maak de routes aan via `app.config()`.
   - Maak een navigatiemenu voor je site, waarin je naar `#/viewnaam` verwijst voor de diverse views.
   - Zie voor een demo `09_ng_route.html` en de verwante bestanden.

b) Breid je app uit met verschillende extra pagina's (dus: views). De werkwijze is in het algemeen als volgt:
   - Maak de view met gewenste gegevens in de map `/views`.
   - Maak een controller voor de view, of breidt je bestaande `controllers.js` uit (als je alle controllers in één bestand bundelt).
   - Breidt je route-tabel uit.
   - Breid de hoofdnavigatie op je homepage uit. Test de uitbreiding.

c) Maak een master/detailview, waarbij gebruik wordt gemaakt van `$routeParams`. Let op de volgende zaken:
   - Pas de HTML in de masterview aan, zodanig dat de `<a href>` de routeparameter meeneemt.
   - Pas de routing table in je `.config()` aan, zodanig dat de parameter wordt herkend (bijvoorbeeld `:id`).
   - Pas de detailcontroller aan, zodanig dat de `$routeParams` worden opgepikt en op basis daarvan worden gebonden in de user interface (bijvoorbeeld `$routeParams.id`).

- Zie eventueel de routing-table in `js/09_app.js` voor inspiratie.

05 – FACTORIES EN SERVICES

a) Voeg een factory toe aan je applicatie en laat deze de gegevensvoorziening verzorgen. Hanteer globaal de volgende werkwijze:
   - Voeg een `.factory()` toe aan je AngularJS-module. Laat de factory een object retourneren, waarin je de data-retrieval methods exposed.
   - Injecteer de `factory [naam]` in je controller.
   - Gebruik de exposed get-methode uit de factory om de gegevens op te halen.
   - Zie bijvoorbeeld `10_ng_factory.html` als voorbeeld.

b) Doe hetzelfde, maar dan via een Service.

c) Gebruik een AJAX-call in je factory om live data op te halen.
   - API's zijn bijvoorbeeld beschikbaar op `openweathermap.org/API` of `www.omdbapi.com`.
   - Injecteer `$http` in je factory, om XHR-calls te kunnen doen.
   - Roep de methode aan in je controller, let er op dat `$http` een promise teruggeeft. Gebruik dus de callbacks `.success()` en/of `.then()` om de chain af te handelen.
   - Let er op dat het responseobject (of: dataobject) dat door `.success()` en door `.then()` wordt teruggegeven verschillend is! Het object dat `.then()` teruggeeft is chainable (en bevat daarom andere inhoud). Het object dat door `.success()` wordt teruggegeven is een 'eindproduct' en bevat alleen de data.

d) Schrijf een eenvoudige CRUD-applicatie. Zorg er voor dat de Factory (of Service) alle data-beheer doet.

e) Voeg een CONSTANT toe aan je app. Zet/refactor hierin bijvoorbeeld de url waar je service naar toe gaat, of plaats hierin publieke, veelgebruikte variabelen (`appName`, `appVersion`).
   - Gebruik `angular.module('mijnapp').constant('…', {…});`
   - Maak een keuze tussen CONSTANT en/of VALUE

06 – INTERCEPTORS

a) Breid je module uit met een interceptor. Schrijf bijvoorbeeld een tekst in de console, elke keer als een interceptor wordt afgevuurd.
   - Zie eventueel `13_ng_interceptor.html` als voorbeeld.
   - Denk er aan dat de interceptor in feite een factory is.
   - Vergeet niet de interceptor op de `$httpProvider` te pushen!

b) Maak een interceptor die een event afvuurt. Vang dit event op in de controller die op dat moment in scope is.
   - Denk aan het injecteren van `$scope` in de controller. Ook als je controllerAs-syntaxis gebruikt! (voor event handling is `$scope` nog steeds nodig. ControllerAs kent hier (nog?) geen alternatief voor).
   - Gebruik `$scope.$on()` om events af te vangen en te verwerken.

c) Zorg er voor dat de interceptor data meestuurt (bijvoorbeeld de URL, een timestamp, of data die je zelf verzint). Hergebruik deze data in de controller die het event opvangt.
   - Zie eventueel `13_ng_interceptor02.html` voor een voorbeeld.

07 – DIRECTIVES

a) Breid je pagina uit met een aantal default directives die zijn besproken. Denk bijvoorbeeld aan:
   - `ng-show`, `ng-hide`, `ng-class`, `ng-change`, `ng-cloak` en meer.
   - Zie de directives-documentatie online voor voorbeelden

b) Custom directives: breid je module uit met een custom directive. Begin bijvoorbeeld met een directive die de huidige datum & tijd in de pagina toont.
    - Bepaal of de directive een Attribuut of Element of beide kan zijn.
    - Schrijf een template voor de app.
    - Voeg de directive in via je HTML-code
c) Schrijf een directive die inhoud kan uitlezen uit attributen van de directive en verwerkt in de pagina.
d) Schrijf een directive met een `link: function(){…}` waarin de inhoud van de directive wordt bewerkt. Geef de tekst bijvoorbeeld een ander lettertype of achtergrondkleur.
    - Gebruik de jqLite-mogelijkheden die worden genoemd op docs.angularjs.org/api/ng/function/angular.element.
e) Maak een eigen directive die iets nuttigs doet. Leg de werking ervan uit aan een collega.
f) Zoek op http://ngmodules.org/ naar directives. Gebruik er een of meerdere in je app.
g) Maak een begin met het lezen van de Custom Directives Series van Dan Wahlin, op http://weblogs.asp.net/dwahlin/creating-custom-angularjs-directives-part-i-the-fundamentals . Probeer deze concepten toe te passen.

07 - COMPLETE APPS

a) Lees de blogpost over AngularJS-bronnen op http://www.kassenaar.com/blog/post/2014/02/20/Bronnen-voor-AngularJS.aspx. Volg een van de tutorials onder 'Het totaalplaatje'.
    - Een leuk voorbeeld om te volgen is *A Step-by-Step Guide to Your First AngularJS App*. Hier zijn aparte Angular modules die in de hoofdmodule worden geinjecteerd het uitgangspunt.
    - Bekijk ook de andere links en volg de apps die hier worden gemaakt.
b) Maak een eigen app met AngularJS voor je werk, hobby of familie/gezin. Dit zijn de specs:
    - De app heeft een homepage en minimaal 2 views.
    - Een van de views is een detailview op basis van een keuze die de 'master'view is gemaakt (vergelijk Yindo: in de eerste view kies je een boek, in de tweede view lees je details over dat boek).
    - De app bevat minimaal één custom directive.
    - De app is opgezet volgens de aanbevolen AngularJS-architectuur met modules, controllers en een factory die voor gegevensvoorziening zorgt.
    - De factory mag statische data serveren als je geen gebruik maakt van een remote API.

08 – AUTHENTICATIE

a) Bekijk het voorbeeldproject over client-sided authenticatie in AngularJS en bestudeer de volgende elementen:
    - Hoe de constanten in de app worden gedefinieerd en waarvoor ze dienen (in `app.js`).
    - Bekijk hoe de routing wordt uitgebreid met custom properties.
    - Bekijk de Authorization Service en Session service en beredeneer waar ze voor dienen.
    - Bestudeer de login-form en –controller. Bekijk met name het gebrek aan 'eigen' logica en hoe de taken gedelegeerd worden aan de Services en het zenden van events (met `$broadcast()`).
    - Bekijk de functie van de application controller en het verwerken van de events gaat (via `$scope.$on()`).
    - Bestudeer tot slot de interceptors en probeer een totaalplaatje te vormen hoe het inlog-proces in zijn werk gaat.
b) Maak een eigen project met client-sided inloggen.
    - Indien je geen API hebt voor inloggen, kun je alles client-sided bouwen/mocken.

- Als je inlogt tegen een 'echte' service/API, zorg er dan voor dat je weet welke data de API retour stuurt en hoe je deze via de interceptors in de requestheaders verwerkt.