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 hiervoor als demo `03_ng_filter.html`.
   - Breid je demo uit met enkele aanvullende, ingebouwde filters zoals `uppercase`, `lowercase` of `orderby`.
   - Zie voor documentatie http://docs.angularjs.org/api/ng/filter.

02 – CONTROLLERS

a) Wijzig je demo, zodanig dat de data nu door een controller wordt geleverd. De HTML-code (en werking) blijft dan gelijk, maar op de achtergrond komt het resultaat op andere wijze tot stand.
   - Zie als voorbeeld `05_ng_controller01.html` en `05_ng_controller02.html`.
   - De controller staat (voorlopig) in de global scope. Later gaan we dit omzetten naar een nettere module-notatie.

b) 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.

c) 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`.

03 – MODULES

a) Gebruik je eigen demo en refactor de code, zodanig dat de controller nu *uit* de global JavaScript-scope wordt gehaald, en wordt toegekend aan een AngularJS-module.
   - Denk er aan de directive `ng-app` in de HTML uit te breiden met een modulenaam.
   - Maak een `angular.module()` op basis van de modulenaam.
   - Voeg de controller toe aan je module.
   - Zie voor een demo bijvoorbeeld `07_ng_module01.html`.

b) 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.
   - Breidt je module uit met een tweede controller. Onderzoek of je in de tweede controller binnen dezelfde module bij gegevens uit de `$scope` van de eerste controller kunt komen.
   - Zie voor een demo `08_ng_module02.html`.

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_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.
   - Gebruik de console en eventueel Batarang (als Chrome-plug-in) voor debugging.

05 – FACTORIES

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_controller.html` als voorbeeld.

b) Gebruik een AJAX-call in je factory om live data op te halen.
   - API's zijn bijvoorbeeld beschikbaar op `openweathermap.org/API` of `ergast.com/mrd/` .
   - 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.

06 – DIRECTIVES

a) Breidt 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

b) Schrijf een directive die inhoud kan uitlezen uit attributen van de directive en verwerkt in de pagina.

c) Schrijf een directive met een `link: function(){…}` waarin de inhoud van de directive wordt bewerkt. Geef de tekst bijvoorbeeld een ander lettertype of achtergrondkleur.

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.