OEFENINGEN ANGULAR 2

01 – EENVOUDIG BEGINNEN

a) Het algemene Github-adres met voorbeeldcode is github.com/PeterKassenaar/voorbeeldenAngular2
b) Maak een eigen Hello World-app. Dit kan op verschillende manieren:
   - Begin met een lege map en voeg zelf `package.json`, `tsconfig.json`, enzovoort toe. De volgorde staat in het bestandje `Angular 2 dependencies.txt`.
   - OF: gebruik als basis `01-helloworld` en vul zelf aan waar nodig.
   - OF: volg zelf de stappen in de QuickStart-demo op `angular.io`.
c) Probeer eens of je een nieuwe component kunt maken en deze bootstrappen. De nieuwe component komt dan *in plaats van* `<hello-world>`, nog niet erbij. Volg deze stappen:
   - Maak een nieuw bestand, bijvoorbeeld \app\nieuwe.component.ts.
   - Importeer de juiste afhankelijkheden.
   - Schrijf een selector en een HTML-template.
   - Pas de bootstrapper aan, zodat daarin nieuwe component wordt geïmporteerd en gestart.
   - Pas index.html aan, zodat de juiste selector wordt gebruikt.
   - Draai `npm start` om je nieuwe component te testen.
d) Installeer Angular-CLI en maak hiermee een basisproject. De aanwijzingen hiervoor vind je op https://cli.angular.io/ . Gebruik de opdrachten `ng new`, `ng serve` en `ng generate`. Zoek zelf op hoe dit werkt.
e) Optioneel, om kennis te maken TypeScript : ga naar `typescriptlang.org` en oefen met het maken van een Class. Maak bijvoorbeeld een klasse `Person`, met properties als `firstName`, `lastName` en `email`. Test hoe deze properties worden bereikt door ze in de console te loggen.
   - Bekijk ook andere voorbeelden uit het dropdown-menu (zoals `Types`, `Generics`, `Modules`) om breder kennis te maken met TypeScript.

02 – DATABINDING

a) Breid je app uit oefening 1 (`Hello World`) uit met een field/propertie. Bindt deze property in het template dat gebruikt wordt.
   - Doe dit zowel impliciet (met rechtstreekse initialisatie van variabelen) als expliciet (met aparte declaratie en initialisatie in de `constructor`).
b) Maak een array van properties. Bind ze in het template met de directive `*ngFor`.
   - Geef via TypeScript expliciet op dat de property uit een array van strings of een array van objecten moet bestaan (maak zelf deze keuze).
c) Maak expliciet een `Model` voor de inhoud van je array. Dat mag een object met één of meerdere properties zijn. Zorg er voor dat de inhoud van de array bestaat uit objecten van het type `<Model>`.
   - Gevorderd: in plaats van een class te gebruiken voor het `Model` mag je ook het TypeScript-construct `Interface` gebruiken. Zoek eventueel zelf op hoe dit werkt. Conventie is dat in dat geval als naamgeving `<IModel>` wordt gebruikt.
d) Bedenk zelf een toepassing voor de directive `*ngIf` en gebruik deze in combinatie met je class en properties.
e) Breng de template naar een externe .html-file. Gebruik vervolgens in `@Component` de property `templateUrl` om hier naar te verwijzen. Test of de databinding blijft werken.

03 – EVENT BINDING

a) Voeg een element met event-binding toe aan je applicatie. Bijvoorbeeld een `(click)` op een knop afvangen. Roep een event handler in de component aan als de event optreedt. Laat deze bijvoorbeeld een `alert()` tonen of een `console.log()`-melding.
   - Zie voor voorbeelden eventueel de map `\03-eventbinding` in de voorbeeldbestanden.

b) Test meerdere typen DOM-events, bijvoorbeeld `blur`, `focus`, `keypress`, `mousemove`, enzovoort. Zie eventueel Mozilla.org (https://developer.mozilla.org/en-US/docs/Web/Events) voor een overzicht.

c) Maak een formulierveld (bijvoorbeeld een tekstveld) met een local template variable. De notatie hiervoor is `#myFieldName`. Geef de variabele door aan de event handler en verwerk hem in een `alert()`, `console.log()`-melding, om een waarde in het DOM te updaten, enzovoort.

d) Maak een eenvoudige client-sided CRUD-applicatie: users kunnen elementen toevoegen aan een array (namen, producten, enzovoort) en verwijderen uit de array. Gebruik voor verwijderen de JavaScript-functie `ArrayName.splice(…)`.

04 - ATTRIBUTE BINDING EN TWO-WAY BINDING

a) Maak een component met een tekstvak. Als de user een (Engelstalige) kleur in het tekstvak invult en op een knop klikt, krijgt een bijbehorende deze achtergrondkleur.
   - Breid dit eventueel uit met een tweede tekstvak voor het instellen van de tekst/voorgrondkleur.
   - Gevorderd: onderzoek hoe dit werkt als de kleuren uit een serie radio buttons gekozen kunnen worden, of uit een selectielijst/dropdown.

b) Optioneel: onderzoek zelf hoe de Angular-concepten *class binding* en *style binding* worden gebruikt. Aanwijzingen hiervoor staan op https://angular.io/docs/ts/latest/guide/template-syntax.html#!#other-bindings.

c) Maak een tekstveld in je component met two-way binding. De syntaxis hiervoor is `[(ngModel)]`. Toon de waarde van de ingevoerde tekst in de pagina.
   - Maak een kopieerfunctie: Maak twee tekstvakken op de pagina. Tekst die in het ene tekstvak wordt ingevuld, verschijnt ook in het andere tekstvak.

05 - SERVICES

a) Als het goed is heb je inmiddels enkele componenten met hierin diverse data. Verplaats deze data vanuit de component naar een service.
   - Injecteer de service in de hoofdcomponent waarin je de data wilt gebruiken.
   - Denk aan de property `[providers]` in de `@Component`-annotatie.

b) Gebruik de alternatieve notatie en importeer je service *app-wide* (in `boot.ts`). Test of alles nog net zo werkt als voorheen.

c) Werken met async-services: maak zelf een .json-bestand met data en laadt dit in je applicatie. Denk onder meer aan:
   - het toevoegen van `http.dev.js` aan index.html;
   - het injecteren van `Http` in de service;
   - het aanpassen van de component waarin de service wordt geconsumeerd: `HTTP_PROVIDERS` injecteren en de observable-notatie met `.subscribe()` gebruiken.

d) Meer over RxJs: importeer deze library in je component met `import 'rxjs/Rx';` en gebruik andere methoden. Test bijvoorbeeld:
   - `.map()`, `.filter()`, `.delay()`, `.retry()` en andere.
   - Een voorbeeld staat in `\07-services-rxjs`.
   - Bekijk enkele video's op https://egghead.io/technologies/rx (gebruik je headset) en pas deze technieken toe in je Angular-applicatie.

e) Werken met Live API's: gebruik een API naar keuze om live data op te halen. Maak een nieuwe applicatie en gebruik hierin deze data.
    - API's zijn bijvoorbeeld beschikbaar op `openweathermap.org/API` of `www.omdbapi.com`.
    - Zie ook het bestand `JavaScript APIs.txt` voor meer API-endpoints.

06 - APPLICATIES MET MEERDERE COMPONENTEN

a) Maak een nieuwe applicatie, of ga uit van je applicatie uit de eerdere oefeningen.
b) Voeg een detail-component toe. De details worden getoond na een muisklik op de hoofdcomponent.
    - Volg het stappenplan (4 stappen) uit de presentatie, waarin de component wordt gemaakt, ingesloten in de hoofdcomponent en op de juiste wijze in HTML wordt aangeroepen.
    - Een voorbeeld staat in `\10-components`.
c) Gebruik de annotatie `@Input()` om data vanuit de hoofdcomponent door te geven aan de detailcomponent.
    - Zorg er voor dat een object wordt doorgegeven aan de detailcomponent. Gebruik hiervoor de notatie `[objName]="objData"` in de template van de hoofdcomponent.
    - Toon de inhoud van het object in de detailcomponent.
d) Gebruik de annotatie `@Output()` om events vanuit de detailcomponent door te geven aan de hoofdcomponent. Denk bijvoorbeeld aan het doorgeven van een rating (zoals in het voorbeeld), of iets anders.
e) Complete applicatie: maak een eenvoudige eCommerce-applicatie met de volgende onderdelen:
    - Een Store. Schrijf een service voor de store die een 4 of 5 producten levert (dit mag statische data zijn).
    - Een Detailcomponent. Bij klikken op een product worden details over het product getoond.
    - Een Shopping Cart. De user kan het product in een winkelmandje plaatsen. De inhoud van het winkelmandje is steeds zichtbaar in de hoofdcomponent.
    - Een "Order"-knop. Als hierop wordt geklikt, wordt de inhoud van het winkelmandje getoond en het totaalbedrag berekend.

07 - ROUTING

a) Maak je applicatie uit de vorige oefeningen geschikt voor routing, of begin een nieuwe applicatie. De eisen zijn de volgende:
    - Zorg er voor dat de applicatie een rootComponent heeft, waarin het "hoofdmenu" met opties wordt getoond.
    - Zorg er voor dat verschillende componenten worden geladen als een keuze wordt gemaakt in het hoofdmenu.
    - Volg het stappenplan (8 stappen) uit de presentatie, bekijk eventueel map `\13-router` voor een voorbeeld.
b) Maak je applicatie geschikt voor het werken met route-parameters. Denk bijvoorbeeld aan het maken van een detailroute voor een bepaald product in je applicatie. Denk aan:
    - het uitbreiden van `@RouteConfig` met parameters;
    - het insluiten van `RouteParams` om de parameters te kunnen oppikken in de detailcomponent;
    - Het aanpassen van de HTML om parameters te kunnen meegeven (`[routerLink]=…`)
    - In de detailcomponent gegevens opvragen op basis van de meegegeven parameter.
c) Maak je applicatie geschikt voor 2 of 3 parameters. Onderzoek zelf hoe dit gaat.
    - In Angular 1 kon je optionele parameters gebruiken, zoals `/:id?`. Kan dit in Angular 2 Router ook? Zo ja, implementeer dit. Zo nee, wat is een mogelijke workaround?

d) Gebruik een van de lifecycle hooks. Test bijvoorbeeld de werking van `routerCanActivate` en/of `routerCanDeactivate`. Maak een kleine component waarin deze werking wordt gedemonstreerd.