IONIC WORKSHOP – WERKINSTRUCTIES

Dit zijn de instructies voor de Ionic Workshop, waarin de volgende stappen worden doorlopen:

- Deel 1: Installeren en aanmaken van basis Ionic App
- Deel 2: Navigatie met behulp van het Ionic Menu
- Deel 3: Navigatie met behulp van Ionic Tabs
- Deel 4: Navigatie via de NavController

Ieder onderdeel van de workshop kent een *doel* en een set aan *instructies*, waarin de stappen staan omschreven om een steeds grotere Ionic applicatie te bouwen. Daarnaast staan aan het eind van ieder onderdeel een aantal *handige links* waar meer informatie gevonden kan worden, en mogelijke instructies voor een *extra opdracht*, voor degene die snel klaar zijn met de instructies en die de app nog verder willen uitbouwen.

DEEL 1: INSTALLEREN EN AANMAKEN VAN BASIS IONIC APP

Doel:

Ionic en Cordova (en eventueel nodejs) installeren, en de Ionic CLI gebruiken om een eerste applicatie op te zetten.

Instructies:

Om een eerste Ionic app te maken, is het makkelijk de Ionic CLI te gebruiken, waarmee de app snel kan worden opgezet en lokaal kan worden gedraaid.

> *Let op: om Ionic en de CLI te installeren zijn node en npm nodig. Als deze nog niet op je computer beschikbaar zijn, download dan eerst hier node:* https://nodejs.org/en/

Als node en npm opgezet zijn, kunnen Ionic en Cordova geïnstalleerd worden. Installeer beide packages globaal (*-g*), door de volgende regel uit te voeren op de command line:

```
npm install -g ionic cordova
```

Als Ionic geïnstalleerd is, kan de Ionic CLI gebruikt worden om een eerste app te bouwen. In deze workshop wordt de meest standaard template gebruikt (*blank*) om mee te beginnen. Alternatieve templates hebben al een veel uitgebreidere structuur, en bevatten bijvoorbeeld al menu's (*sidemenu*) of een tab-navigatie (*tabs*), maar voor nu volstaat de meest 'kale' versie.

Navigeer in de command line naar een folder waar je de app wilt plaatsen, en voer daarna het volgende commando uit:

```
ionic start myApp blank
```

Hier zorgt *ionic start* dat je app aangemaakt wordt, is *myApp* de naam van je applicatie (je kan hier een willekeurige andere naam verzinnen), en geeft *blank* aan welke Ionic template wordt gebruikt (alternatieve keuzes zijn o.a. *tabs*, *tutorial*, *sidemenu* of *conference*).

Nu de basis voor de eerste app er staat, kan de Ionic CLI gebruikt worden om de app te bouwen en lokaal te draaien. Navigeer daarvoor naar de zojuist gemaakte folder, en voer het Ionic commando *serve* uit. Op de command line:

```
cd myApp
ionic serve
```

Het commando *serve* zorgt ervoor dat de app wordt gebouwd (nieuw aangemaakte bestanden komen in de folder www), en in je browser komt te draaien onder localhost port 8100 (andere port nummers, bijvoorbeeld 1234, kunnen worden gekozen door het commando *ionic serve -p 1234* uit te voeren). De CLI luistert ook naar wijzigingen die in de code worden aangebracht; alle wijzigingen die bijvoorbeeld in de `src` folder worden gemaakt zullen meteen worden gebouwd in de `www` folder, en daarmee ook meteen weer in de browser te zien zijn.

Handige Links:

De Installation Guide van Ionic: https://ionicframework.com/docs/intro/installation/

De opties voor standaard Ionic templates: https://ionicframework.com/docs/cli/starters.html

DEEL 2: NAVIGATIE MET BEHULP VAN HET IONIC MENU

Doel:

Een nieuwe pagina binnen de app toevoegen, en een menu gebruiken (`<ion-menu>`) om naar deze nieuwe pagina te kunnen navigeren.

Instructies:

Nu de app is opgebouwd en lokaal draait, kan een nieuwe pagina worden toegevoegd, inclusief een standaard Ionic menu waarmee navigatie binnen de app mogelijk wordt.

De eerste stap is om de nieuwe pagina aan te maken. Maak binnen de folder 'pages' een nieuwe folder, en noem deze 'about'. Maak in deze folder een nieuw 'about.html' en een 'about.ts' bestand aan.

```
> src
    > pages
        > home
        > about
            about.html
            about.ts
```

In het HTML bestand wordt later nog content en een menu header toegevoegd. In het about.ts bestand kan nu alvast de basis van het component worden toegevoegd:

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

@Component({
  selector: 'page-about',
  templateUrl: 'about.html'
})
export class AboutPage {
    constructor (public navCtrl: NavController) {}
}
```

Nu het nieuwe component is aangemaakt, dient deze ook toegevoegd te worden aan de app module. Importeer het nieuwe component, en voeg het toe aan de declarations en entryComponents lijsten in app.module.ts:

```
import { AboutPage } from '../pages/about/about';
...
@NgModule({
    declarations: [
        MyApp, HomePage, AboutPage
    ],
    ...
    entryComponents: [
        MyApp, HomePage, AboutPage
    ]
})
```

De volgende stap is om de nieuwe 'about' pagina binnen app.component.ts beschikbaar te maken, omdat vanuit hier de navigatie tussen de pagina's mogelijk moet worden gemaakt. Naast het importeren van een aantal nieuwe modules en componenten, moeten in het app.component de verschillende pagina's worden gedefinieerd, en dient een `openPage` methode gemaakt te worden, welke wordt aangeroepen wanneer van de ene naar de andere pagina wordt genavigeerd. Voeg eerst de volgende imports toe aan app.component.ts:

```
import { Component, ViewChild } from '@angular/core';
import { Platform, MenuController, Nav } from 'ionic-angular';

import { AboutPage } from '../pages/about/about';
```

Vervolgens moeten een instantie van de NavController en een lijst met pagina's worden geïnstantieerd. Voor de NavController instantie wordt gebruik gemaakt van Angular's `@ViewChild`. In `class MyApp` binnen app.component.ts, voor de constructor:

```
@ViewChild(Nav) nav: Nav;
rootPage: any = HomePage;
pages: Array<{title: string, component: any}>;
```

En daarna kunnen in de constructor de pagina's en de Ionic MenuController worden gedefinieerd:

```
constructor (public menu: MenuController, platform: Platform,
                statusBar: StatusBar, splashScreen: SplashScreen) {
    platform.ready().then(...);
    this.pages = [
        { title: 'home', component: HomePage },
        { title: 'about', component: AboutPage }
    ];
}
```

De laatste stap die in het app.component.ts bestand moet worden genomen is het aanmaken van de `openPage` methode, die wordt gebruikt nadat een gebruiker in het menu een van de twee pagina's selecteert. Die selectie wordt meegegeven aan de `openPage` methode, welke ervoor zorgt dat de geselecteerde pagina de nieuwe `rootPage` wordt, en dat het menu weer wordt gesloten. Maak de volgende methode aan in `class MyApp` binnen app.component.ts:

```
openPage(page): void {
    this.nav.setRoot(page.component);
    this.menu.close();
}
```

Met deze logica kunnen de verschillende pagina's worden geopend; de volgende stappen zijn om een `<ion-menu>` element toe te voegen in het app.html bestand, en om aan de html's van de Home en About pagina's een knop toe te voegen waarmee het menu daadwerkelijk kan worden geopend.

Het `<ion-menu>` element krijgt een `<ion-header>` met daarin een titel, en een `<ion-content>` blok met daarin de links naar de twee pagina's. Voeg de volgende code toe in app.html:

```
<ion-menu [content]="content">
    <ion-header>
        <ion-toolbar>
            <ion-title>Menu</ion-title>
        </ion-toolbar>
    </ion-header>
    <ion-content>
        <ion-list>
            <button ion-item *ngFor="let page of pages"
                    (click)="openPage(page)">
                {{page.title}}
            </button>
        </ion-list>
    </ion-content>
</ion-menu>
<ion-nav #content [root]="rootPage"></ion-nav>
```

Het menu heeft een property binding `[content]`, waar een string als parameter aan wordt meegegeven; deze string (in dit geval 'content', maar dat kan ook een willekeurige andere string zijn) moet gelijk zijn aan de variable declaration `#content` op het `<ion-nav>` element.

Het menu is nu gereed en heeft een verwijzing naar de twee pagina's (`*ngFor="let page of Pages"`) en kan gebruikt worden om deze pagina's te openen (`(click)="openPage(page)"`). Het menu is echter nog niet zichtbaar in de app; hiervoor moet op de Home en About pagina's een knop worden toegevoegd, waarmee het menu daadwerkelijk kan worden geopend (met behulp van de Ionic directive `menuToggle`). Voeg aan de home.html en about.html bestanden de volgende knop in de headers toe:

```
<ion-header>
    <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title>About</ion-title>
    </ion-navbar>
</ion-header>
```

Met deze knop kan het menu geopend worden, en kan er naar de twee verschillende pagina's genavigeerd worden. Hiermee is dit deel van de workshop nu afgerond.

Extra opdracht:

Met de Ionic CLI is het mogelijk om snel nieuwe componenten, directives en pagina's te maken. Dit kan met het commando `ionic generate <type> <name>`. Dit commando kan makkelijk toegepast worden om in de app een nieuwe contact pagina te maken, welke vervolgens aan het menu toegevoegd kan worden. Voer de volgende regel in op de command line:

```
ionic generate page Contact --no-module
```

De optionele --no-module parameter zorgt ervoor dat er voor het nieuwe component geen NgModule wordt gegenereerd.

In de `src` folder zullen nu een contact- .ts, .scss en .html file zijn toegevoegd. Om deze Contact pagina aan het menu toe te voegen, dient de pagina eerst toegevoegd te worden aan de app module. Vervolgens kan de pagina in *app.component.ts* worden toegevoegd aan de lijst met pagina's:

```
this.pages = [
    { title: 'home', component: HomePage },
    { title: 'about', component: AboutPage },
    { title: 'contact', component: ContactPage }
];
```

Door het toevoegen van de Contact pagina aan de lijst met pagina's, wordt het al direct mogelijk om naar de nieuwe pagina te navigeren via het menu. Het enige wat nog ontbreekt is content op de pagina, en de mogelijkheid om terug te navigeren naar de andere twee pagina's; voeg hiervoor in contact.html eenzelfde Ionic header toe zoals in de Home en About pagina's worden gebruikt:

```
<ion-header>
    <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title>Contact</ion-title>
    </ion-navbar>
</ion-header>
<ion-content padding>
    <p>
        Zoek je contact? Mail dan wiewilcontactmetmij@zoeenzaam.nl
    </p>
</ion-content>
```

Nu is het wel weer mogelijk om van iedere pagina het menu te openen, en naar iedere andere pagina te navigeren.

Handige links:

Ionic Menu API: https://ionicframework.com/docs/api/components/menu/Menu/

Meer details over het CLI commando *ionic generate*: https://ionicframework.com/docs/cli/generate/

Een korte Angular 2+ guide gemaakt door de ontwikkelaars van Ionic: http://learnangular2.com/

DEEL 3: NAVIGATIE MET BEHULP VAN IONIC TABS

Doel:

Een nieuwe pagina binnen de app toevoegen, en tabs gebruiken (`<ion-tabs>`) om naar deze nieuwe pagina te kunnen navigeren.

Instructies:

Naast navigeren met behulp van het `<ion-menu>` element, worden in Ionic apps ook vaak tabs gebruikt om mee te navigeren. In dit deel van de workshop worden tabs toegevoegd aan de Home pagina, waarmee naar een nieuwe pagina genavigeerd kan worden. De nieuwe pagina wordt een lijst met items, waar verderop in de workshop nog meer functionaliteit aan toegevoegd zal worden.

Om navigatie met tabs voor elkaar te krijgen, moeten 2 nieuwe componenten aangemaakt worden; een nieuw component genaamd 'List' (waarin een simpele lijst met items wordt getoond), en een component genaamd 'Tabs' (waarbinnen een verwijzing komt naar de Home en List pagina's).

Maak eerst het component 'List' aan, in een nieuwe map binnen de *pages* folder. In het typescript bestand (*list.ts*) wordt een `itemList` gemaakt, welke in de constructor wordt gevuld met items:

```
@Component({
    selector: 'page-list',
    templateUrl: 'list.html'
})
export class ListPage {
    itemList: Array<{title: string, note: string, icon: string}>;

    constructor() {
        this.itemList = [
            {title:'Item1', note:'this is item #1', icon:'wifi'},
            {title:'Item2', note:'this is item #2', icon:'beer'},
            {title:'Item3', note:'this is item #3', icon:'alarm'},
            {title:'Item4', note:'this is item #4', icon:'basket'}
        ]
    }
}
```

In de bijbehorende HTML (*list.html*) moet dezelfde `menuToggle` button komen te staan als op de Home pagina, zodat het menu vanuit beide pagina's beschikbaar blijft:

```
<ion-header>
    <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>
        <ion-title>List</ion-title>
    </ion-navbar>
</ion-header>
```

Naast de header kan in een `<ion-content>` element de lijst getoond worden die eerder in de component constructor is aangemaakt. In deze lijst wordt nu nog een button gebruikt, welke pas in het volgende deel van de workshop met meer functionaliteit uitgebreid zal worden. Voeg voor nu alleen de volgende code toe aan *list.html*:

```
<ion-content>
    <ion-list>
        <button ion-item *ngFor="let item of itemList">
            <ion-icon name="{{item.icon}}"></ion-icon>
            {{item.title}}
            <span class="item-note" item-right>
                {{item.note}}
            </span>
        </button>
    </ion-list>
</ion-content>
```

Vergeet niet het nieuwe component ook toe te voegen aan de app module (in app.module.ts). Het tweede component dat nodig is, is het 'Tabs' component, waarmee navigatie door middel van tabs mogelijk wordt. Maak een nieuwe folder 'tabs' aan (ook binnen de *pages* folder), en maak daarin het typescript bestand *tabs.ts* aan:

```
import { Component } from '@angular/core';

import { HomePage } from '../home/home';
import { ListPage } from '../list/list';

@Component({
    templateUrl: 'tabs.html'
})
export class TabsPage {
    tab1Root = HomePage;
    tab2Root = ListPage;

    constructor() {}
}
```

In dit component worden de Home en List pagina's geïmporteerd, en worden deze vervolgens toegewezen aan de variabelen `tab1Root` en `tab2Root`. Deze variabelen kunnen nu worden gebruikt in de HTML, met een `<ion-tab>` element. Voeg de volgende code toe aan het nieuwe bestand tabs.html:

```
<ion-tabs>
    <ion-tab [root]="tab1Root" tabTitle="Home" tabIcon="home">
    </ion-tab>
    <ion-tab [root]="tab2Root" tabTitle="List" tabIcon="contacts">
    </ion-tab>
</ion-tabs>
```

De `<ion-tab>` elementen verzorgen alle navigatie naar de Home en List pagina. De laatste stap om dit deel van de workshop af te maken, is om in het bestand app.component.ts de verwijzing naar de Home pagina te vervangen door een verwijzing naar de nieuwe Tabs pagina. Hiervoor moet de import van HomePage vervangen worden door een import van TabsPage (vergeet ook niet TabsPage toe te voegen aan app.module.ts), en moet TabsPage in de lijst met pagina's worden toegevoegd:

```
import { TabsPage } from '../pages/tabs/tabs';
...
this.pages = [
    { title: 'home', component: TabsPage },
    { title: 'about', component: AboutPage }
];
```

Navigatie binnen de app is nu mogelijk via het menu (navigeren naar Home en About), en via de tabs die vanuit de Home pagina beschikbaar zijn gemaakt (navigeren naar Home en List).

Extra opdracht:

Naast de verschillende manieren die Ionic biedt om naar pagina's te navigeren, waar deze workshop vooral veel aandacht aan besteedt, kent Ionic ook een grote verscheidenheid aan componenten die specifiek ontworpen zijn voor gebruik op mobiele apparaten. In deze extra opdracht zullen een paar van die componenten gebruikt worden, om de List pagina met nieuwe functionaliteit uit te breiden.

De List pagina zal worden uitgebreid met een optie om nieuwe items aan te maken en toe te voegen aan de lijst. Eerst wordt een toggle button toegevoegd waarmee de input velden getoond of verborgen kunnen worden. Daarna worden twee input tekst velden getoond (voor een item naam en een item omschrijving), en een selection component, waarmee een item icon kan worden gekozen (namen van iconen zijn specifiek, vandaar dat het geen input veld is maar een selection veld). Tenslotte wordt een button toegevoegd waarmee het nieuwe item in de lijst kan worden opgenomen.

Plaats eerst boven de lijst met items in *list.html* een toggle button, die bepaalt of de input velden worden getoond:

```
<ion-content>
    <ion-item>
        <ion-label>Allow adding items</ion-label>
        <ion-toggle [(ngModel)]="allowAdding"></ion-toggle>
    </ion-item>
    <ion-list>
        ...
    </ion-list>
```

```
        <div *ngIf="allowAdding">
            <p>Hier komen zo de input velden</p>
        </div>
    </ion-content>
```

De onderaan de pagina wordt nu alleen getoond wanneer de gebruiker de toggle button aanzet. In deze kunnen nu twee tekst input velden worden toegevoegd, waarmee naam en omschrijving kunnen worden opgegeven. Deze input velden komen in een <ion-list> element te staan, met een <ion-title> element daarboven:

```
<div *ngIf="allowAdding">
    <ion-title>Add new item</ion-title>
    <ion-list>
        <ion-item>
            <ion-label fixed>Name</ion-label>
            <ion-input [(ngModel)]="name" type="text"></ion-input>
        </ion-item>
        <ion-item>
            <ion-label fixed>Description</ion-label>
            <ion-input [(ngModel)]="note" type="text"></ion-input>
        </ion-item>
    </ion-list>
</div>
```

Vervolgens kan onder de text input velden nu ook een <ion-select> element worden toegevoegd, waarmee een keuze voor een icoon kan worden gemaakt:

```
<ion-item>
    <ion-label>Icon</ion-label>
    <ion-select [(ngModel)]="icon"
                submitText="ok"
                cancelText="cancel">
        <ion-option value="paper-plane">
            Paper plane
        </ion-option>
        <ion-option value="football">
            Football
        </ion-option>
        <ion-option value="logo-angular">
            Angular logo
        </ion-option>
        <ion-option value="attach">
            Paperclip
        </ion-option>
    </ion-select>
</ion-item>
```

Nu de input velden zijn toegevoegd, kan ook een Ionic button worden geplaatst waarmee het nieuwe item in de lijst opgeslagen kan worden. Plaats de button net onder de <ion-list>:

```
<div padding>
    <button ion-button (click)="addItem()">Add item</button>
</div>
```

De laatste stap is om de methode `addItem` toe te voegen in *list.ts*, zodat het item daadwerkelijk aan de lijst kan worden toegevoegd, en de input velden kunnen worden leeggemaakt:

```
export class ListPage {
    name: string;
    note: string;
    icon: string;

    ...

    addItem(): void {
        if (this.name && this.note && this.icon) {
            this.itemList.push({
                title: this.name,
                note: this.note,
                icon: this.icon
            });
            this.name = this.note = this.icon = null;
        }
    }
}
```

De methode controleert nu of alle velden ingevuld zijn voordat een item wordt toegevoegd, en doet niets wanneer dat niet het geval is. De app kan eventueel nog verder uitgebreid worden door een Ionic `alert` component toe te voegen, waarmee de gebruiker gewaarschuwd kan worden over welke velden nog ontbreken. Lees hier hoe een `alert` gebruikt kan worden op de Ionic Docs pagina:

https://ionicframework.com/docs/components/#alert

Handige links:

Overzicht van Ionic iconen: https://ionicframework.com/docs/ionicons/

API van Ionic Tabs: https://ionicframework.com/docs/api/components/tabs/Tabs/

Overzicht van alle Ionic componenten: https://ionicframework.com/docs/components/#overview

DEEL 4: NAVIGATIE VIA DE NAVCONTROLLER

Doel:

Een nieuwe pagina aan de app toevoegen, en deze met behulp van de NavController naar de top van de 'stack' te schuiven.

Instructies:

In dit deel van de workshop wordt een *ItemDetails* pagina toegevoegd, waar de gebruiker heen kan vanaf de *List* pagina door op een item op de lijst te klikken. Hierbij wordt gebruik gemaakt van Ionic's NavController, welke een array aan pagina's bewaart. Deze array kan gezien worden als een stapel, waarop nieuwe pagina's kunnen worden toegevoegd (push), of waarvan pagina's kunnen worden verwijderd (pop); de bovenste pagina van de stapel zal altijd getoond worden in de applicatie.

Om dit voor elkaar te krijgen moet een nieuw component gemaakt worden (item-details), welke naar de top van de stapel 'gepushed' kan worden zodra de gebruiker op een item in de lijst klikt. Maak een folder *item-details* aan, en maak daarbinnen een *item-details.ts* en *item-details.html* aan.

```
@Component({
    selector: 'page-item-details',
    templateUrl: 'item-details.html'
})
export class ItemDetailsPage {
    constructor() {}
}
```

Deze nieuwe pagina kan bovenop de stapel geplaatst worden vanuit het *list* component. Daarvoor dient eerst in de html een `(click)` event toegevoegd te worden aan de lijst met items. Wijzig het `<button>` element in *list.html* als volgt:

```
<button ion-item
        *ngFor="let item of itemList"
        (click)="openItem(item)">
```

Vervolgens kan de methode `openItem` toegevoegd worden aan het component, waarin de nieuwe *ItemDetails* pagina daadwerkelijk bovenop de stapel geplaatst kan worden. Plaats in *list.ts* de NavController als parameter in de constructor, en voeg de methode `openItem` toe waarin de NavController wordt gebruikt:

```
constructor (public navCtrl: NavController) { ... }

openItem(item): void {
    this.navCtrl.push(ItemDetailsPage, {
        item: item
    });
}
```

De `push()` methode van NavController kent een verplichte parameter *page* (wat een string of een pagina kan zijn, in dit geval de pagina *ItemDetailsPage*), en een optionele parameter *params* (wat een object moet zijn, waar in dit geval het item-object uit de `itemList` aan wordt meegegeven). Deze laatste parameter kan worden uitgelezen met behulp van Ionic's NavParams, wat gebruikt kan worden om in de *ItemDetails* pagina daadwerkelijk de details van het item te tonen. Voeg in het component van *ItemDetails* een `selectedItem` object toe, en voeg de NavParams toe aan de constructor waarmee de details van het meegegeven item uitgelezen kunnen worden:

```
selectedItem: any;

constructor(public navParams: NavParams) {
    this.selectedItem = navParams.get('item');
}
```

Tenslotte kan het `selectedItem` object daadwerkelijk in de html getoond worden. In *item-details.html*:

```
<ion-header>
    <ion-navbar>
        <ion-title>
            Item Details
        </ion-title>
    </ion-navbar>
</ion-header>
<ion-content>
    <div *ngIf="selectedItem">
        <h3 text-center>
            {{selectedItem.title}}
            <ion-icon name="{{selectedItem.icon}}"></ion-icon>
        </h3>
        <h4 text-center>
            Display details for: <b>{{selectedItem.title}}</b>
        </h4>
    </div>
</ion-content>
```

In de `<ion-header>` is deze keer geen knop voor het menu opgenomen; Ionic heeft er zelf voor gezorgd dat in het menu een icon komt te staan waarmee de gebruiker kan terug navigeren naar de pagina waar hij/zij vandaan kwam.

Extra opdracht:

Het Ionic framework biedt een uitgebreide mogelijkheden om de styling van een applicatie snel te wijzigen om de app een custom uiterlijk te geven. In dit laatste extra deel van de workshop wordt een nieuw component aan de Home pagina toegevoegd, waarbij vervolgens verschillende styling attributen geconfigureerd kunnen worden.

Voeg eerst de volgende componenten toe (een `<ion-range>` component binnen een `<ion-card>` component) aan *home.html*:

```
<ion-card>
    <ion-card-header>
        How awesome is Ionic:
    </ion-card-header>
    <ion-item>
        <ion-range min="0" max="10" ([ngModel)]="fun">
            <ion-icon range-left name="sad"></ion-icon>
            <ion-icon range-right name="happy"></ion-icon>
        </ion-range>
    </ion-item>
    <ion-card-content>
        This awesome: {{fun}}/10
    </ion-card-content>
</ion-card>
```

In het `<ion-range>` component, net als in veel andere componenten, komt de Ionic 'primary color' terug, wat in dit geval blauw is (schuif de slider heen en weer om de kleur te zien). Ionic maakt gebruik van meerdere standaard kleuren. Probeer het `range` component bijvoorbeeld eens een andere kleur mee te geven:

```
<ion-range min="0" max="10" ([ngModel)]="fun" color="secondary">
```

De waarde van de `primary`, `secondary` en andere kleuren worden bepaald in het *variables.scss* bestand, in de folder *theme*. Pas hier de waarde van de kleuren aan, om de app bijvoorbeeld een meer typisch Ordina uiterlijk te geven:

```
$colors: (
    primary: #e98300,
    ...
);
```

In dit bestand kunnen veel meer kleuren worden toegevoegd om de app een unieke stijl mee te geven. Daarnaast kunnen hier custom style templates geïmporteerd worden, zoals nu ook de Ionic iconen hier al worden ingeladen.

Ionic kent ook een aantal CSS attributen die makkelijk toegepast kunnen worden op de componenten, omdat ze zo vaak gebruikt worden. Denk hierbij bijvoorbeeld aan tekst allignment of *padding* en *margin* van elementen (padding is in deze workshop al op sommige plaatsen van de app gebruikt, zoals op het `<ion-content>` component in *home.html*). Deze Ionic-CSS attributen zijn erg snel toe te passen; voeg het attribuut `float-right` toe aan het component <ion-card-content>, en het component zal daarmee de styling krijgen waarbij het rechts op de pagina wordt uitgelijnd:

```
<ion-card-content float-right>
    This much fun: {{fun}}/10
</ion-card-content>
```

Op de Ionic pagina zijn alle CSS Utilities terug te vinden waarmee de styling van de app makkelijk en snel kan worden aangepast (check de handige links hieronder).

Handige links:

De API van de NavController: https://ionicframework.com/docs/api/navigation/NavController/

Ionic CSS Utilities: https://ionicframework.com/docs/theming/css-utilities/

Details over Ionic Theme variabelen: https://ionicframework.com/docs/theming/theming-your-app/