**Bouwen van Mobile apps met Angular en NativeScript**

Steeds meer Java ontwikkelaars komen de laatste jaren in aanraking met front-end ontwikkeling. Voor deze groep ontwikkelaars is het nog maar een kleine stap om tot mobiele app ontwikkeling over te gaan middels NativeScript. Dit artikel is bedoeld voor ontwikkelaars die kennis hebben van JavaScript en/of Angular en dus gebruik kunnen maken van de voordelen van NativeScript.

Maar wat is NativeScript eigenlijk? En waarom zou je NativeScript gebruiken?

**Wat is NativeScript?**

NavtiveScript is een open source framework (onder de Apache 2 licence) om native IOS en Android apps te bouwen, met behulp van JavaScript en CSS. NativeScript is een andere technologie dan de hybride-frameworks, zoals Ionic. NativeScript is een runtime, geen web technologie. Je app zal niet draaien als een mini website in een WebView en is daarom meer performant. Met NativeScript heb je direct toegang tot al de Native APIs van je device.

**Waarom NativeScript?**

* Een van de argumenten om NativeScript te gebruiken is hergebruik van ’Skills’. Dat wil zeggen iemand met kennis van JavaScript kan direct aan de slag met NativeScript. NativeScript wordt namelijk geschreven in JavaScript.
* Ten tweede hergebruik van code. Schrijf Native Mobile apps voor iOS en Android met één enkele code base. Het is niet zo vreemd dat de slogan van NativeScript luidt: ‘Write ones, run everywhere’.
* Het is tevens gemakkelijk uit te breiden met behulp van NativeScript modules (zie de voorbeelden in dit artikel) en npm modules.
* Tenslotte wordt er geen gebruik gemaakt van WebViews, zoals bij hybride frameworks. Middels JavaScript hebt je direct toegang tot de Native APIs en is daardoor beter performant.

**Wat is er benodigd voor NativeScript?**

Om aan de slag te gaan met NativeScript dien je Node.js te installeren en vervolgens installeer je de NativeScript CLI met het volgende commando:

npm install –g nativescript

Om snel aan de slag te gaan met NativeScript, is het verstandig om ‘NativeScript Sidekick’ te installeren.

Als je met NativeScript Sidekick ervoor kiest om je apps te builden in de cloud, dan kan je onafhankelijk van je operating system waarop je werkt, je apps ontwikkelen. Je kan dan uiteraard zelfs iOS apps builden op een Windows machine. Een andere optie is om lokaal te werk te gaan, maar daarvoor moet je je eigen omgeving inrichten met iOS Xcode en Android SDK.

**Hoe werkt NativeScript met behulp van JavaScript**

Hieronder volgt een simpel voorbeeld van JavaScript in NativeScript, die een Objective-C based IOS UIAlertView control instantieert:

var myAlert = new UIAlertView();

myAlert.message = “NativeScript rocks!”;

myAlert.show();



Omdat web developers geen iOS en Android specifieke APIs willen aanleren, biedt NativeScript een set van NativeScript modules aan. Die modules abstraheren de iOS en Android details in simpele JavaScript APIs.

De bovenstaande UIAlertView-gebaseerde code kan herschreven worden met de NativeScript ‘Dialog module’:

var dialogs = require("ui/dialogs");

dialogs.alert({ message: "NativeScript rocks!" });

Deze dialogs.alert()call levert ons ook de [android.app.AlertDialog](http://developer.android.com/reference/android/app/AlertDialog.html) voor je Android app.

Ondanks dat dit ‘dialog’’ voorbeeld eenvoudig is, kan dezelfde techniek gebruikt worden voor het bouwen van robuuste apps, door gebruik te maken van de reeds bestaande en volwassen native iOS en Android UI componenten.

**Wat kan Angular toevoegen aan NativeScript?**

NativeScript kan nu ook geschreven worden in Angular.

Als je kennis hebt van Angular, dan is het een kleine stap om Angular in NativeScript te gebruiken.

Het grote verschil met Angular is dat de browser-based HTML-elementen zoals <div> en <span> niet beschikbaar zijn in NativeScript. Je dient daarvoor in de plaats NativeScript UI componenten te gebruiken.

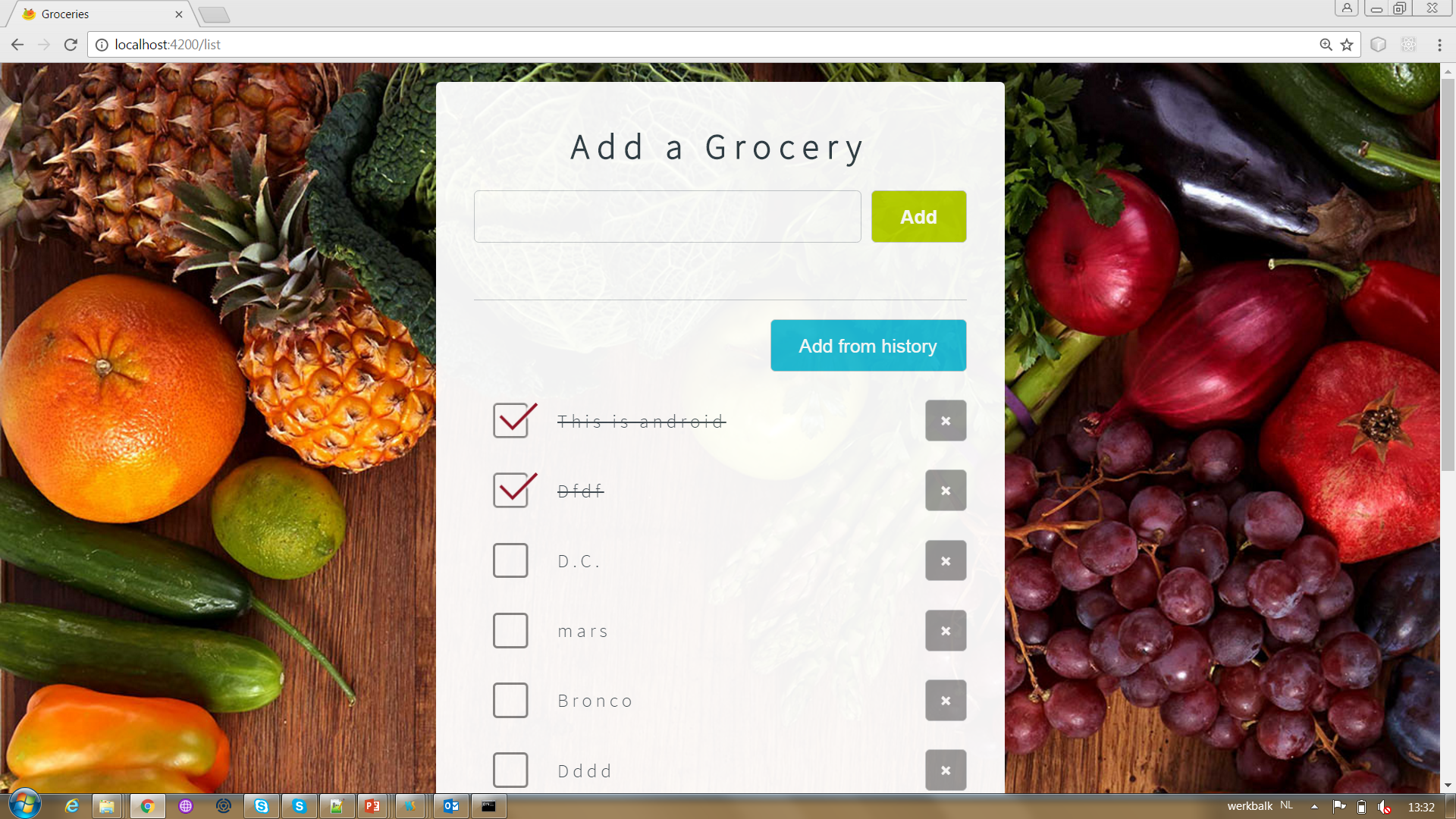
In NativeScript wordt geen DOM of browser gebruikt. NativeScript UIs zijn native UIs en dus losgekoppeld van de DOM. Omdat Angular een agnostisch framework is en losgekoppeld is van de DOM, kan dit framework gemakkelijk geïntegreerd worden met NativeScript. Onderstaand voorbeeld gaat hierop in.

*AngularJS* is in tegenstelling tot Angular niet geschikt voor NativeScript, omdat dit framework gekoppeld is aan de DOM.

Door gebruik te maken van Angular in NativeScript, heb je de mogelijkheid om code te delen tussen je bestaande webapplicatie en je Native apps. Laten we kijken naar een voorbeeld.

**Hergebruik van code tussen web en mobile apps**

Laten we als voorbeeld een ‘Grocery List’ (nederlands: boodschappenlijst) tonen in een webapplicatie.

****

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

@Component({

selector: ‘grocery-list'

templateUrl: ‘grocerylist.template.html'

})

export class GroceryListComponent {

groceries: string[];

constructor() {

this.groceries =[‘grocery1’, ‘grocery2’]

}

}

Listing 1

In listing 1 is een Angular Component gedefinieerd, die een array van ‘groceries’ vult, via de constructor.

Er wordt gebruik gemaakt van TypeScript. Dit is een ‘typed superset’ van JavaScript en is de standaard voor het schrijven van Angular applicaties.

**Hoe werkt dit component?**

Via de *selector* behorend bij het Component, kan je Angular vragen om dit component te instantieren en te renderen, waar het een <grocery-list> - tag vindt in de HTML (zie listing 2)

….

<grocery-list></grocery-list>

….

Listing 2

Voor het renderen van de ‘Grocery list’, dient er tenslotte een template gedefinieerd te worden met de naam: **grocerylist.template.html** (zie listing 3). Deze template staat ook gedefinieerd onder de *templateUrl* van het Component.

<p>Groceries: </p>

<ul>

<li \*ng-for = “let grocery of groceries">

{{grocery}}

</li>

</ul>

Listing 3

Er wordt door de lijst met ‘groceries’ geïtereerd, met behulp van de *ng-for* directive van Angular.

*De vraag is nu:* wat moeten we anders programmeren om bovenstaande code werkend te krijgen voor je NativeScript iOS en Android apps?

Het uitgebreide antwoord is:

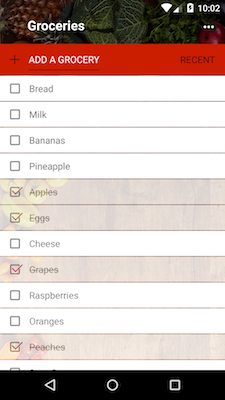
Het component in listing 1, blijft hetzelfde. Deze TypeScript-code kan je dus hergebruiken tussen de webapplicatie en mobile apps.

Het enige wat je dient te wijzigen is *de template* voor dit component!

Je dient NativeScript UI componenten te gebruiken in plaats van HTML-elementen in je template (zie listing 4).

Wat opvalt is dat je nog steeds de *ng-for* directive van Angular kan gebruiken voor het itereren door de lijst.

En het omzetten van een template naar NativeScript UI componenten is een eenvoudige exercitie, omdat er voor ieder HTML-element wel een Native UI component te vinden is. Dit voorbeeld laat ook zien dat Angular losgekoppeld is van de DOM, en dus met NativeScript UI componenten kan omgaan.

****

<StackLayout \*ngFor = "let grocery of groceries">

<Label [text] = “grocery"></Label>

</StackLayout>

Listing 4

**HTTP Module**

Een ander voorbeeld voor code hergebruik is het volgende.

NativeScript biedt support voor web APIs, zoals **XMLHttpRequest en fetch().**

Je kan hiervoor de HTTP module gebruiken binnen NativeScript.

Deze module maakt het mogelijk om web requests te verzenden en web responsies te ontvangen.

Listing 5

var http = **require**(“**http**”); 🡪 **import HTTP module**

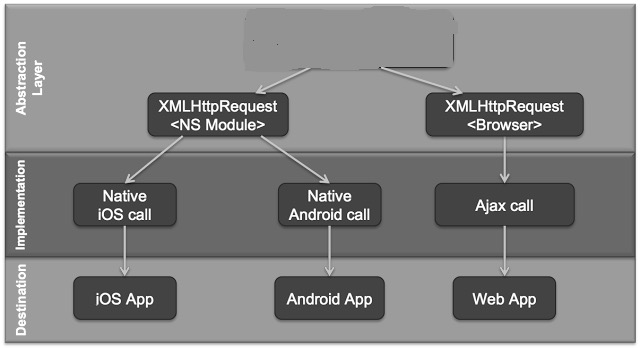
http.getJSON(“https://api.myservice.com”)

.then(function (result)) {

…

});

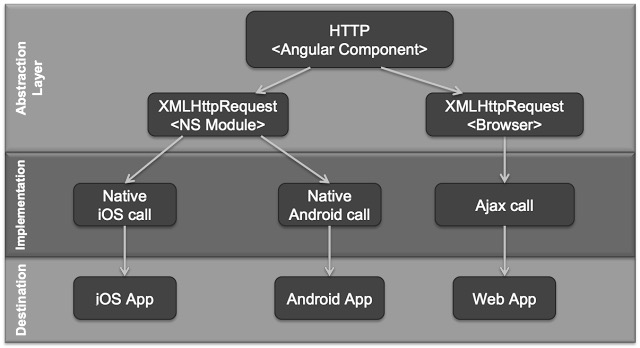
In onderstaand schema zie je aan de linkerzijde dat NativeScript de **XMLHttpRequests web API** vertaalt naar Native code voor iOS en Android. Tevens wordt aan de rechterzijde van dit schema de implementatie van de **XMLHttpRequest** getoond, die gebruikt wordt voor de webapplicatie. Dit is een andere implementatie dan de NativeScript variant.



Het probleem hier is dat we de **XMLHttpRequest** implementaties niet kunnen hergebruiken tussen de webapplicatie en mobiele app. Er is hier een ‘’missing link’’, namelijk een generieke HTTP-module voor de webapplicatie en de mobiele app.

Dit kan opgelost worden met Angular. Angular voegt een extra abstractie laag toe voor het afhandelen van HTTP-requests/responses. Dit kan met de HTTP-module van Angular.

Deze module kunnen we hergebruiken tussen de webapplicatie en de mobiele app.



**Tot slot**

Hergebruik van code is een zeer belangrijke uitdaging voor development in het algemeen.

Op dit moment kan je met de Angular CLI een webapplicatie genereren. En voor NativeScript gebruik je de NativeScript CLI. Dit is een probleem, omdat de CLIs ons geen mogelijkheid geven om één project te maken voor de webapplicatie en de Native applicatie.

Natuurlijk is het mogelijk om twee aparte projecten te onderhouden en de gedeelde files tussen de twee projecten te ‘copy/pasten’.

Dit kan ook gelost worden door bijvoorbeeld het volgende ‘seed project’ gebruiken: https://github.com/TeamMaestro/angular-native-seed. Hiermee kan je met één code base je webapplicatie en NativeScript onderhouden.

Samenvattend, NativeScript is een krachtig framework om native cross-platform mobiele applicaties te bouwen met Angular, TypeScript of JavaScript. En je hebt direct toegang tot de Native platform API’s.

NativeScript werkt anders dan hybride frameworks zoals Ionic. Maar is uitermate geschikt voor webdevelopers met kennis van JavaScript en Angular. Het is beter performant en eenvoudig te leren.