

Progressive Web App

Mijn project zoals ik al meerdere keren heb beschreven gaat over een factuur generator tool waar er online gemakkelijk een factuur gemaakt kan worden en kan uitprinten.

Mijn oorspronkelijke idee vanaf het begin was om een applicatie te maken voor de iPhone geheel in Swift.

Swift is een relatief nieuwe programmeer taal van Apple. Met Swift worden zowel MacOS als iOS applicaties gemaakt. De iPhone applicatie had de functies om facturen, offerten en meer documentatie aan te maken. Om een begin te maken wou ik dit realiseren door tutorials te volgen. Ik heb op Udemy, een online tutorials marketplace, een tutorial gekocht dat ging over Swift en hoe je een applicatie kon maken. Zo heb ik langer dan 3 weken elke dag de tutorial gevolgd. Helaas duurde dit steeds maar langer en moesten wij, de studenten, al een eerste opzet inleveren over waar de project over ging.

Ik heb hiermee zitten worstelen en meerdere docenten tips gevraagd over hoe ik dit probleem kan aanpakken, want ik zat er mee dat ik halverwege de tutorials nog steeds niets uit mijzelf kon programmeren in Swift. De docenten die ik gesproken had, hebben mij allen in eigen woorden verteld dat ik het maar moest laten zitten en doorgaan met mijn project, want anders kwam ik in de knoei met de deadline.

Dit heb ik gedaan en mij werd aangeraden om een progressive web app te maken. Hiervoor moest er een basis gemaakt worden en dat is gewoon in HTML met CSS.

Ik ben gaan kijken naar wat een progressive web app eigenlijk is. Dit bleek simpel gezegd een omhulsel te zijn van een website. Zo kon je altijd je gebruikers garanderen dat zij altijd de laatste versie hebben van je product. Ik kreeg ook een aantal voorbeelden te zien en toen drong het tot mij door dat het inderdaad Progressive web apps waren.

Voorbeelden zoals AliExpress en Visual Studio Code waren een van de meest bekende onder ons.

Ik ben onderzoekend te werk gegaan en heb al snel voordelen en wat matige nadelen kunnen vinden over het verschil tussen een progressive web app en een native applicatie, in dit geval dus Swift.

Progressive Web App VS Native Swift			
Voordelen	Nadelen	Voordelen	Nadelen
HTML & Libs	Hoog batterij verbruik	Laag batterij verbruik	Dure productiekosten
Lage productie kosten	Geen 3e partij interactie	Push notificaties	Vaak debuggen
Offline gebruik	Weinig hardware gebruik	Geen hardware limiet	Vaak apps updaten
Eenmalig downloaden	Weinig tutorials	3e partij interactie	Lange leertijd
Geen update installaties	Zwak voor power user	OS exclusiviteit	
Gemakkelijk te leren		Geavanceerde opties	

Gekeken naar deze voor- en nadelen en de deadline was het voor mij nogal logisch dat ik voor de progressive web app ging. Ik kwam er wel achter dat het internet echter weinig progressive web app tutorials bood. Hierdoor kwam ik niet verder dan met een handje vol websites en youtube filmpjes dat net niet genoeg waren, outdated of puur bedoeld is voor Android mobiele apparaten.

Ik ging hier niet alle tijd in besteden dus ik heb gekeken naar wat de basis was van een progressive web app en dat was een website maken puur in HTML. Dit vond ik natuurlijk helemaal geweldig.

Wat ik vaak gelezen had is dat een progressive web app altijd een manifest bestand heeft. Dit is in JSON. Dit heb ik aangemaakt en de daarbij nodige variabelen aan toegevoegd.

Laten we eerst beginnen om een lijstje te maken van wat er nodig is om een progressive web app te maken.


- Manifest.json bestand
- Service Workers
- Progressive web app library
- Firebase database

De manifest is een bestand waarin data verwerkt zit over de hele applicatie. Dit wordt gebruikt om de applicatie te installeren op je mobiele apparaat. Hiermee kun je ook applicaties downloaden zonder dat het vanuit de App Store komt. Dit bestand specificeert de naam van de applicatie, de URL, kleuren en iconen. Hieronder zie je voor een gedeelte hoe de manifest.json eruit ziet van Mantax

```
1  {
2    "name": "MANTAX",
3    "short_name": "MANTAX",
4    "theme_color": "#e31a5c",
5    "background_color": "#ffffff",
6    "display": "fullscreen",
7    "orientation": "portrait",
8    "Scope": "/",
9    "start_url": "/",
10   "icons": [
11     {
12       "src": "/supportingfiles/app-images/images/icons/icon-72x72.png",
13       "sizes": "72x72",
14       "type": "image/png"
15     },
16     {
17       "src": "/supportingfiles/app-images/images/icons/icon-96x96.png",
18       "sizes": "96x96",
19       "type": "image/png"
20     },
21     {
22       "src": "/supportingfiles/app-images/images/icons/icon-128x128.png",
23       "sizes": "128x128",
24       "type": "image/png"
25     },
26   ]
27 }
```

Een service worker is een script in JavaScript dat de gebruikte browser draait op de achtergrond. Google Chrome, Chromium, Firefox en sinds kort Safari ondersteunen deze script. De script activeert de mogelijkheid om push notificaties, achtergrond synchronisatie en offline mogelijkheid van je applicatie. Je installeert als het ware een script in je website code en de browser leest dit en draait het op de achtergrond. Het nadeel hiervan is dat dit ook wordt uitgevoerd als je zelfs niet de browser gebruikt, vandaar al dat hoge energie verbruik. Service workers zijn niet verplicht, want tegenwoordig hebben JavaScript libraries hun eigen manier van een service worker.

Om een duidelijk inzicht te kijken over wat allemaal ondersteund is om een progressive web app te bouwen, kan er simpel gekeken worden op een website genaamd What Web Can Do Today. Op dit website wordt de hardware en operating systeem gescanned van je mobiele apparaat en de resultaten ervan worden direct getoond. Het hangt vaak van de browser af of functies ondersteund zijn, maar de de operating systeem speelt ook een rol.



What Web Can Do Today

Can I rely on the Web Platform features to build my app?
An overview of the device integration HTML5 APIs

✓ Feature available in your current browser ✗ Feature not available in your current browser

Camera & Microphone

- AUDIO & VIDEO CAPTURE ✓
- ADVANCED CAMERA CONTROLS ✗
- RECORDING MEDIA ✗
- REAL-TIME COMMUNICATION ✓

Operating System

- OFFLINE STORAGE ✓
- FILE ACCESS ✓
- CONTACTS ✗
- SMS ✗
- STORAGE QUOTAS ✗
- TASK SCHEDULING ✗

Advertisement

Surroundings

- BLUETOOTH ✗
- USB ✗
- NFC ✗
- AMBIENT LIGHT ✗

Input

- TOUCH GESTURES ✗
- SPEECH RECOGNITION ✗
- CLIPBOARD (COPY & PASTE) ✓
- POINTING DEVICE ADAPTATION ✓

Dit was dan het voorbereidende stukje. Nu over naar het praktiserende stuk.

Een progressive web app maken kan met verschillende frameworks. Om er een paar te noemen zijn er bijvoorbeeld: React, Polymer, Webpack, Knockout, Lighthouse, Accelerated Mobile Pages, Ionic 2 en als laatst AngularJS.



React wordt beheerd en ondersteund door Facebook die het ook op hun eigen website gebruiken. Je kunt dus al merken dat React een zeer sterk framework, want dit wordt dagelijks hevig getest door 1.18 miljard gebruikers.

Ook is React de fundering van React Native. React Native maakt progressive web apps mogelijk te porten naar een native applicatie.

Een aantrekkelijke voordeel van ReactJS is dat het in componenten werkt tijdens het bouwen van de applicatie. Elke component wordt gecodeerd in JavaScript en je kunt het gemakkelijk hergebruiken, want daar draait het uiteindelijk om bij "lazy coding". Nog een voordeel is dat er dus in JavaScript gecodeerd wordt. Hierdoor hoeft je als programmeur geen nieuwe taal te leren en kan data snel door de DOM-laag gaan.



Polymer Projects heeft een grote voordeel en dat is dat je een groot verschil van tijd kunt besparen door een van hun progressive web app templates te gebruiken. Dit is een open source project van Google en wordt frequent geupdate met templates.

De Polymer templates gebruiken een PRPL patroon om de installatie van de applicatie naar de apparaat te optimaliseren. Polymer kun je gebruiken als je snel een werkende prototype wilt hebben zodat je gelijk door kunt met het ontwerpen van je applicatie.



awesome
webpack

Webpack is een module bundelaar. Dat wilt zeggen dat Webpack modules JavaScript met afhankelijkheden en maakt ze statisch zodat ze ingebouwd worden en niet meer erop moet wachten via een internet verbinding.



Vaak heb je helemaal geen uitgebreide framework nodig zoals React om een progressive web app te bouwen. Je kunt Knockout gebruiken om lichte applicaties te ontwerpen. Daarmee bedoel ik applicaties die niet zwaar zijn voor de hardware van de apparaat.

Knockout gebruikt JavaScript om de Model-View-ViewController te verbinden. Zo komt het met meer voordelen zoals dat je de library van kunt toevoegen aan je progressive web app zonder alles te hoeven herschrijven zodat het werkt, de library is maar 13 KiloBytes klein en terwijl het zo klein is komt het met verassend veel functionaliteit.

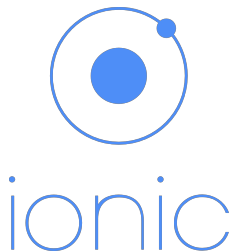
Knockout wordt door veel programmeurs gezien als **de** framework voor kleine projecten waarbij tijd en snelheid van programmeren prioriteiten hoog liggen. Knockout wordt ook vaak aangeraden, omdat het gemakkelijk te leren is voor beginners in tegenstelling tot React.



Een handige plugin voor je progressive web app is Lighthouse. Dit is een hulpmiddel dat de performance monitort van je applicatie. Het wordt als een plugin geïnstalleerd in Google Chrome en via de plugin kun je je eigen progressive web app bezoeken. Tegelijk wordt je performance bijgehouden en gerapporteerd.



Angular de nieuwe versie van AngularJS is een relatief nieuwe framework en meteen al een van de beste. Deze staat aan kop samen met ReactJS. Het is een wat zwaardere framework dan React en dat betekent dat er veel meer functionaliteit in zit dan React zelf. Angular is zeer geprefereerd door het bedrijf Java en .NET programmeurs.



Ionic is een Angular gebaseerde framework dat ook in het zelfde jaar is uitgebracht als Angular. Ionic heeft wel een ander doel en dat is gefocust op moderne responsive web apps. De gebruikers van Ionic komen vooral vanuit de frameworks Cordova en PhoneGap.

Ik zal het zelf hebben over Angular en hoe ik dit zou kunnen implementeren zodat ik een progressive web app kan bouwen uit mijn website.

Eerst creëer ik een nieuw project in Angular en gebruik ik met de plugin NPMJS een command om Angular naar de laatste mogelijke versie te updaten. Ik maak voordat ik dit doe eerst een map aan met de naam van mijn progressive web app en binnen in de map open ik een console/terminal. Nu de update:

NPM I -G @ANGULAR/CLI

Daarna maak ik een project aan in Angular:

NG NEW PWA-MANTAX

pwa-Mantax is de naam van mijn project.

Om verder te gaan moet de library van Angular worden toegevoegd met het project.

NPM I @ANGULAR/PWA@NEXT

Door dit te doen zullen er twee documenten tevoorschijn komen namelijk de manifest.json en de ngsw-config.json. De ngsw-config.json zal zich bevinden in de root van de map en de manifest.json in een nieuw mapje genaamd assets.

Even een korte uitleg over wat de bestand ngsw-config doet:

Dit bestand is de bestand waar alle cache heen gaat en opgeslagen wordt. Cache verkrijgt je door te browsen. Doordat je dit opgeslagen krijgt, de applicatie afsluit en weer opent. Zul je merken dat het direct precies hetzelfde laat zien zonder eerst te laden. Ook maakt dit bestand offline mode mogelijk. Hier kun je ook de instellingen aanpassen van de service workers scripts.

Volgende stap is om angular.json te openen, daarna zoek je de woorden serviceWorker: #. Dit is een boolean en dit zet je op true als dit nog niet true is.

Daarna is het tijd om het offline te testen. Voordat ik het ga testen moet ik de progressive web app wel eerst “bouwen”, dit is het zelfde principe als de “build” functie bij codeer programma’s zoals Xcode or Visual Studios. Dit is geen “run” command. Ik bouw de project met:

NPM RUN BUILD

Zodra dit gelukt is, is het tijd om de server elementen toe te voegen en dit kan met:

NPM I -G HTTP-SERVER

Uiteindelijk kan ik het gaan testen door in de terminal terug te navigeren naar de root van mijn project en dan de command uit te voeren:

HTTP-SERVER

De applicatie is nu in Angular opgestart en om het na te checken dat de applicatie ook offline staat, kan je dat zien bij de developers tools optie van je browser. Daar kun je ook zien wat de server workers doen.

Als laatste om er echt van zeker te zijn dat je een progressive web app hebt in plaats van een responsive browser is door een Install button te plaatsen in je applicatie. Je kunt de installatie knop gewoon in de HTML toevoegen en wanneer dat geklikt wordt speelt er een script af.

```
<BUTTON *NGIF="PWA.PROMPTEVENT" (CLICK)="INSTALLPWA()">INSTALL</BUTTON>
```

Dit stukje hierboven speelt het volgende script af:

```
INSTALLPWA(): VOID {  
  THIS.PWA.PROMPTEVENT.PROMPT();  
}
```

Wat er nu gebeurd moet worden is dat de applicatie dus nieuwe functionaliteit heeft en weer opnieuw gebouwd moet worden met de run build command. Wanneer dat klaar is weer de server hosten door de HTTP-server command te gebruiken.

Nu is de applicatie omgezet naar een echte progressive web app, maar je hebt het nog niet gezien. Om toegang te krijgen online tot de applicatie en de installeer knop te gebruiken moet er nog een ding gebeuren.

Je eigen IP adres van je computer moet je opzoeken en op je telefoon te browser naar 000.000.000:port. Waar 0 staat moeten de cijfers staan van de computer en waar port staat moet de poort van de server dat je net hebt geopend. Dit kun je vinden in de installatie bestand van Angular: Angular.js.

Zodra je een goede tutorial online kunt vinden is het best simpel om een progressive web app te maken, maar heel veel van deze zijn te uitgebreid waardoor het te moeilijk overkomt en waarbij je al gauw de kluts kwijt raakt. Dit stuk wat ik geschreven heb is nog maar heel simpel, maar er komt zeker meer kijken dat dit.