Bachelor’s thesis

Degree programme in Information and Communications Technology

2019

Tuan Doan

# HISTORY AND DEVELOPMENT OF PROGRESSIVE WEB APP

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme in Information and Communications Technology

2019 | 34 pages

Author: Tuan Doan

# HISTORY AND DEVELOPMENT OF PROGRESSIVE WEB APP

The purpose of this thesis was to offer an overview ofthe history of application development, compare application development types, and  discuss how progressive web application became the trending technology.

Mobile applications are favored more compared to web application thanks to smoother user experience and convenience of using a mobile phone. The speed of a browser can not catch up with newest mobile devices. Moreover, user interface of mobile applications is better since it is designed for specific devices. Although the cost of developing mobile application is always higher than that of the web application, developers need to create two versions of the application for it to run on both iOS and Android system. Progressive Web Application technology is the new solution which can combine the strength of both mobile and web application.

This thesis is divided into two main parts: the theoretical and the practical part. In the theory part, every type of application development is introduced. The way how Progressive Web Application works is described in detail along with various supporting technologies and tools. The practical part is an example of a how progressive web application is built from scratch.

The thesis gives an overview of how applications were built in the past and how they are built in the present. Progressive Web application Technology is emphasized as the trend in the industry through various comparisons and information. Last but not least, the practical part of the thesis is used as a way for beginners to approach Progressive Web Application.

KEYWORDS:

Progressive web application, workbox, service worker, app manifest, push notification, HTPPS.

# CONTENTS

## FIGURES

# LIST OF ABBREVIATIONS (OR) SYMBOLS

| | |
|---|---|
| API | Application Programming Interface |
| CDN | Content Delivery Network |
| CSS | Cascading Style Sheets |
| DOM | Document Object Model |
| HTML | Hypertext Markup Language |
| HTTPS | Explanation of abbreviation (Source) |
| PWA | Progressive Web Application |
| JSON | JavaScript Object Notation |
| NPM | Node Package Manager |

# 1 INTRODUCTION

2019 is the blooming year of technology, as people tend to use automatic devices to help with daily works. New technologies are creating a revolution in making life easier. There are many appliances that can operate automatically, such as door, car, cleaning machines and so on. Moreover, other smart devices are also very popular, such as computers, mobile phones, tablets, laptops, etc.

Undoubtedly, the mobile phone is one of the most powerful devices. Its convenience of small size and intelligence have brought a huge advantage, compared to its main rivals such as computers or laptopss. There are more applications that can be installed and used by mobile phone. Moreover, the experience while using those mobile applications is better than using web applications on computer or laptop. It is more compatible, reliable and faster. Yet, developing mobile application is more expensive than web application due to multiple operation systems. One application will need to be built twice, in totally different ways, for both iOS and Android. As for web application, one version would work for all browsers.

Understanding both aspects of developing mobile and web application, a solution called Progressive Web App (PWA) has been researched and promoted. The term PWA was mentioned the first time in 2015 by designer Frances Berriman and Google Chrome engineer Alex Russell to describe an app that utilize the features of new web browsers. It is a combination of mobile app's experience and web app's technology. Both Google and Apple have tried to support the idea with constant developments since 2015. There are several new tools and technologies which have been built during the above mentioned time frame. They are also a great help for PWA to grow stronger.

As PWA is very trending now in technology industry, it would be an interesting topic to be discussed. In this thesis, a news generator application is built for the purpose of demonstration. It can still generate  news of the day even when there is no internet connection, which is one of the competitive edges of PWA. Core technologies for building PWA such as Workbox, service worker etc. will be used.

In this thesis, there are  five chapters:

- Chapter one is the introduction. In this chapter, background of the topic is briefly presented.

- Chapter two introduces application development in general. In this chapter, the history of application development will be discussed. Moreover, all type of application from the past until now will be explained and compared.
- Chapter three is about progressive web application. In this chapter, more information regarding progressive web application will be given. Core technologies and tools are also mentioned.
- Chapter four is an example of how to build a progressive web application from scratch.
- Chapter five is the last chapter which will conclude all issues discussed throughout the thesis.

# 2 APPLICATION DEVELOPMENT IN GENERAL

During the last decade, mobile and web application have been growing rapidly [1] as it has become more and more crucial to a human's daily life. In the past, the mostly used applications were text editors, calculators, calendars, or some simple games. After years of development, the importance of mobile and web application is levitated to a higher level. People utilize applications for navigation, entertainment, management etc. Until now, developers have provided several types of application: native application, web application, and hybrid application. In this section, all mentioned  types of application will be discussed in more detail.

## 2.1 Native application

Native application is an application which is programmed to be used for a certain device or platform. As its name suggests, the word "native" means it is built by using a native language per platform. For example, Java [2], C# [3] are the programming languages for Android and Objective-C [4], Swift [5] are used to make iOS application. There are also some other alternative languages such as C++ [6], Python [7], Kotlin [8].

It is true that each programming language has its own strengths and weaknesses, however, newer ones are slowly surpassing their precedants. For instance, android developers are favoring Kotlin than Java, which has been there since 1995 (Figure 1). The reason is that Kotlin is smarter, faster and safer. [9]

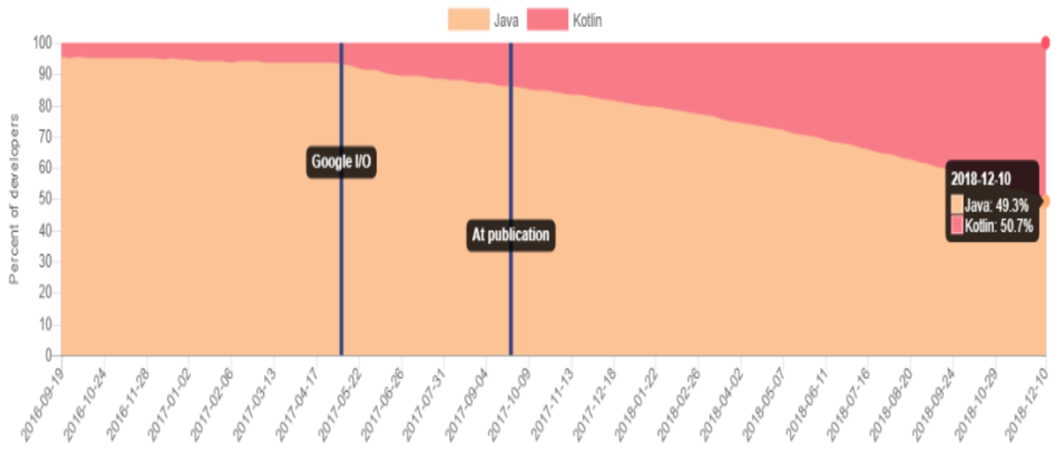


Figure 1. A graph showing a significant increase of Kotlin users [9]

As for the iOS side, the situation is quite the same. Swift, a programming language developed by Apple, is dominating Objective-C. It is easier to use and has higher performance.

Undoubtedly, native applications are becoming more and more popular thanks to theircompatible with mobile devices. They are fast and reliable, since they can be used offline. Another benefit is that there is a wide range of native applications that has same function for users to compare and select the one that is most suitable.

There are two main platforms that developers can publish their applications: Google Play Store and Apple App Store. Either way, developers must accomplish some conditions/tasks (policies, pricing, quality checking etc.) to be able to bring their application to the market. Usually Apple App Store has a stricter requirement to follow than Google Play Store [10]. That is reason why the Apple App Store usually has more quality applications. On the other hand, Google Play Store exceeds at diversity of application. Those compulsory steps such as quality assurance and policy/price checking are quite a challenge for developers when they want to launch an application. However, overall, they are a necessary procedure.

Another issue of developing a native application is that developers need to maintain multiple versions of the same application if they want it to be accessible through different platforms/devices. All in all, it requires a huge effort of work and time. That being mentioned, it might be not always suitable for developers who has limited resources.

## 2.2 Web application

In 1990, Tim Berners-Lee, a physicist at European Organization for Nuclear Research (CERN), successfully created the first web page, using all the components that he built himself: HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML) and the famous WorldWideWeb (WWW).

The very first purposeof a a a website is to display static information, which is available to every user. Ever since, it has been developed constantly and rapidly. Developers have found far more ways to utilize websites: personal websites (blog, online portfolio, curriculum vitae), social networking websites, commercial websites, file-sharing/storing websites and so on. Web application is also included in mentioned list.

Wikipedia defines web application as “a client-server computer program which the client (including the user interface and client-side logic) runs in a web browser”. The word “client-server” is used because web application is created by the combination of client-side script language (Javascript [11], HTML) and server-side script language (Node [12], Java, Python etc.).[13]

Work flow of web application is described as below:

- Web server receives requests from users (through web browser or application UI).
- Requests are forwarded to appropriate web application server.
- Requested tasks are carried out by the web application server.
- Results are sent to the web server.
- Users received the results displayed on web browser or application UI.

It can be seen that web browser and web server are significant factors through out several listed steps. The web browser’s mission is to support and display user interface. On the other hand, the the web server will process and deliver requested data back to users.

The basic skills that developers need for building a web application are HTML, CSS and Javascript. A simple web application can be created using only mentioned tools. Their functionality can be simply described as below:

- HTML is used for marking up texts.

- CSS is used for adding styles the web application.
- Javascript is used for adding functions and making the application more dynamic.

Lately, severalk Javascript frameworks (React [14], Vue [15], Angular [16] etc.)have been launched to ease the process of creating a web application, giving developers various choices at hands. A framework is like a library of premade Javascript code coming with a structure. It not only provides solutions for certain basic problems, but also helps developers to build the skeleton for the project. To be able to use a framework flawlessly, a developer needs to have solid basic Javascript skills first.

Regarding accessibility, web application is not as convenient as native application, since users need to have internet connection to use it. Moreover, user experience will also not be as smooth as that of a native application. On the other hand, a web application does not need to be downloaded and installed to become usable. In addition, one version of a web application can be run on various platforms, which will save developers a great deal of resources.

## 2.3 Hybrid application

A hybrid application is the combination of a native and a web application. It is developed using same technologies as web application: HTML, CSS and Javascript. Then, it is attached into native wrapper such as Apache Cordova or Ionic and displayed through WebView,a tool that helps with rendering HTML.

A hybrid application inherits the ability to work offline and being available within different app stores with just one code base. However, its performance is somewhat inferior compared to native applications. Moreover, it also requires various plugins to access the camera or the the microphone.

# 3 PROGRESSIVE WEB APPLICATION

In this chapter, definition of progressive web app and related technologies will be discussed. Benefits of those technologies and the way to use them will be mentioned.

## 3.1 Definition of progressive web application

Progressive web application is basically web application which offers superior native-like user experience using newest capabilities of modern browsers. According to information on ionicframework.com (website of Ionic - one of the most popular framework to build application), a progressive web application must be:

- Progressive
- Responsive
- Connectivity independent
- App-like
- Fresh
- Safe
- Discoverable
- Re-engageable
- Installable
- Linkable

Any web application can be considered a progressive web application if it can meet all listed requirements. That being said, old web application can become a modern progressive web application with a few upgrades. Developers do not need to build an application from scratch for it to be counted as progressive web application.

## 3.2 Technologies and tools for building progressive web application

Technologies and tools that helps to build progressive web application will be mentioned and discussed in this section.

### 3.2.1 Service worker

A service worker [17] is a separated script file that executes behind the web page, activating additional features (for example, push notification and background sync) without any interaction of user. With the help of service worker, progressive web application can send notification to users and works in offline mode.

As a Javascript worker, it can not manipulate the DOM (Document Object Model, an interface to web pages) directly. An interface called postMessage is used as a connecting bridge, which helps service worker to control the DOM.

One of service worker's uses is handling network request as a programmable network proxy. That being said, a programmer can decide what respond the application will give based on network availability. Therefore, service one of the most crucial factor of progressive web application because of its support for offline experience.

Service worker will be terminated when not in used, and will restart whenever developers need. So, if there is any information that is needed to remain throughout the restart, IndexedDB (a low-level API for client-side storage) can help. Promise (an object in Javascript that will execute a task when certain condition is fulfilled) is used a lots in service worker, so developers would want to know at least the basic of it before diving into service worker.

#### 3.2.1.1 Life cycle of a service worker

Service worker has its own life cycle apart from the web page. The cycle usually starts with installation of service worker. To be able to do that, developers need to check if the current browser supports service worker and then register in the Javascript file of their web page. The installation process will be performed in the background by the browser. Several assets should be cached successfully during the process. Next, an activation event needs to be done for a complete setup. The registration and installation process will be discuss in more details later [Figure 2].

Then, the service worker will be able to handle all the page within its scope. If there is not any message or fetch event coming from user web page, it will move to idle state. After being idle for awhile, service worker will be terminated to save memory.

Below is a figure describing service worker's life cycle:

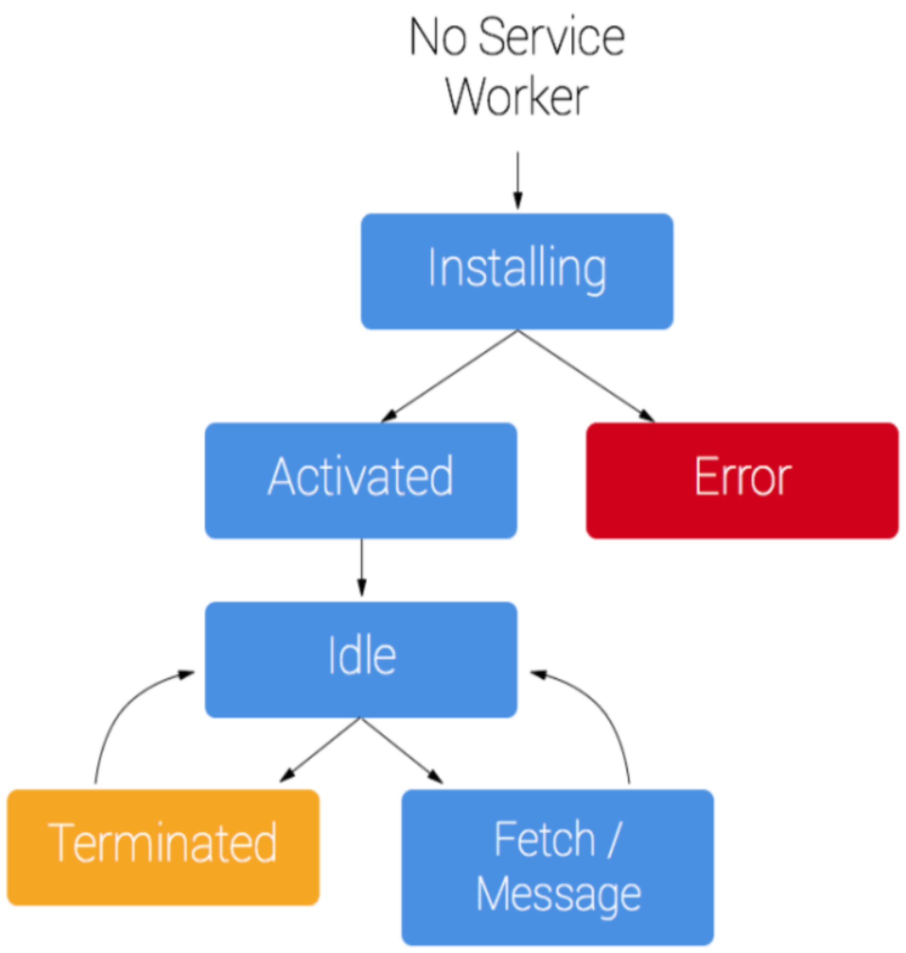
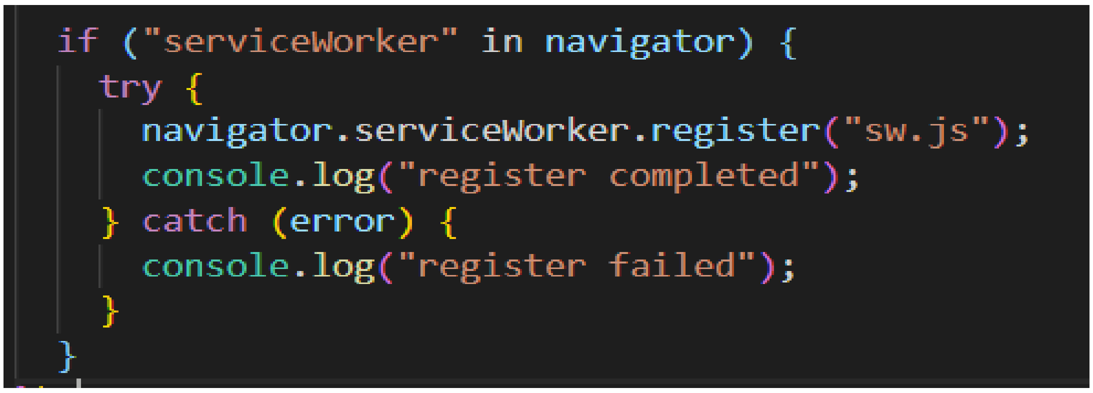


Figure 2. Service worker life cycle [17].

### 3.2.1.2 Register for a service worker

Firstly, developers need to check if the service worker is available in the navigator before the registration step. Then, a condition statement should be created, trying to register for the service worker if it is available, or return an error if it is not.



```
if ("serviceWorker" in navigator) {
  try {
    navigator.serviceWorker.register("sw.js");
    console.log("register completed");
  } catch (error) {
    console.log("register failed");
  }
}
```

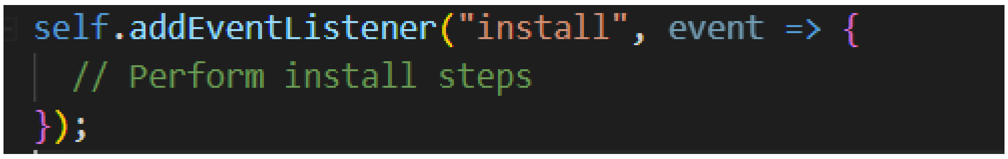
Figure 3. Example of registering for a service worker.

As can be seen from the figure, the console will return either of status strings to inform the developers if the service worker is available or not, and if it is successfully registered or not [Figure 3]. This checking step is important since not every browser support a service worker. If the console returns the string “register completed”, it means that the browser supports service worker, and it has been registered. Otherwise, if it return the other string, developers should check for browser version, or any syntax error in the registering process.

Regarding to the example above, the service worker is registered using a file called sw.js. That is the place where all the instructions for the service worker are written. Service worker is a low level web API and it will not know what to do without any specific commands. The file sw.js should be placed in the root of the project folder, so it can have the control over the whole project.

3.2.1.3 Install a service worker

The next step is installing the service worker. It is usually done within the sw.js file [Figure 4].



```
self.addEventListener("install", event => {
  // Perform install steps
});
```

Figure 4. Example of installing a service worker.

The service worker does not know what to do on it own, so an callback function is added when the event “install” is triggered. In the block of code, there should be installation steps that define which files to be cached. More details will be discussed regarding installation steps later.

#### 3.2.1.4 Cache and return request

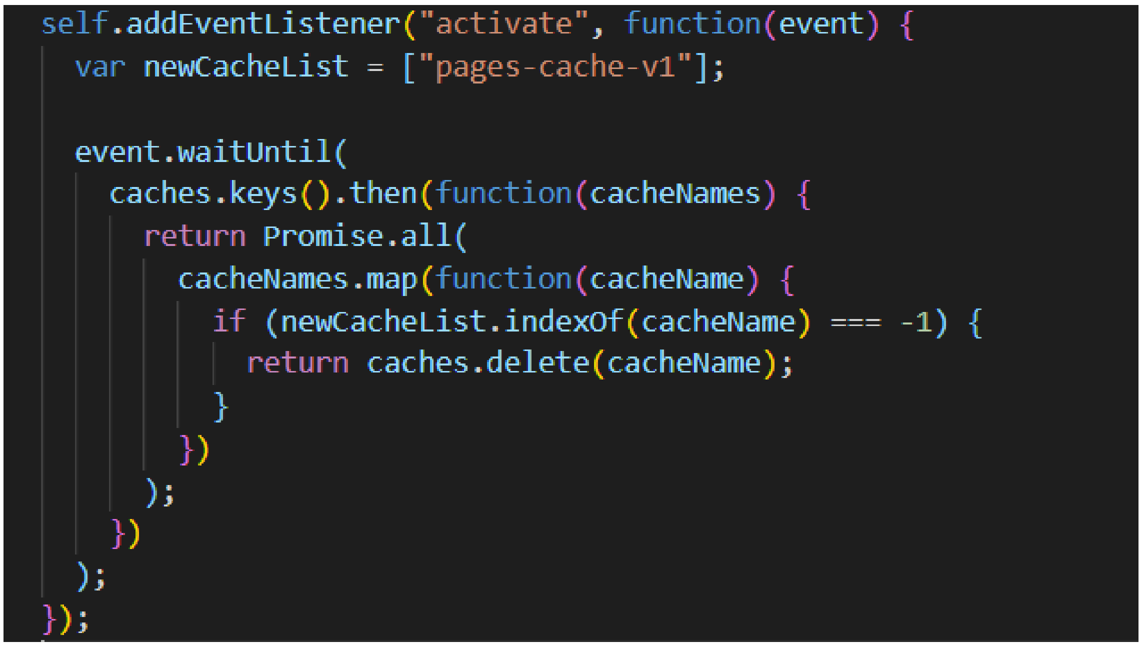
After the installation, a fetch event will be called everytime the page is refreshed or the user redirects to some other pages. What it does is to get the cache that the service worker created. By doing so, progressive web application can show some meaningful contents when there is no internet connection.

Two main functions that are used during this process are event.respondWith() and caches.match(). A promise should be created with the condition is the caches.match() event. If there is any returned data that match the requirement, then it should be returned. Otherwise, a fetch call is returned, trying to get some data from the network. More step by step guide will be introduced later.

#### 3.2.1.5 Update a service worker

As time passes, a service worker will also need to be updated to improve the performance of the application. The process is quite simple: the developers just need to update the service worker JS file. When users try to use the application, the browser will check and see if there is any newer version of the service worker. If there is any changes in the file, installation process of the new service worker will be triggered. But it will not take the control over the old service worker right away until the application is closed. New service worker will be ready for the next use.

Another important thing to note is managing the caches when updating service worker. Developers need to make sure that old caches will be cleared when the new one is activated [Figure 5].



```
self.addEventListener("activate", function(event) {
  var newCacheList = ["pages-cache-v1"];

  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (newCacheList.indexOf(cacheName) === -1) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
```

Figure 5. Example of updating process for a service worker.

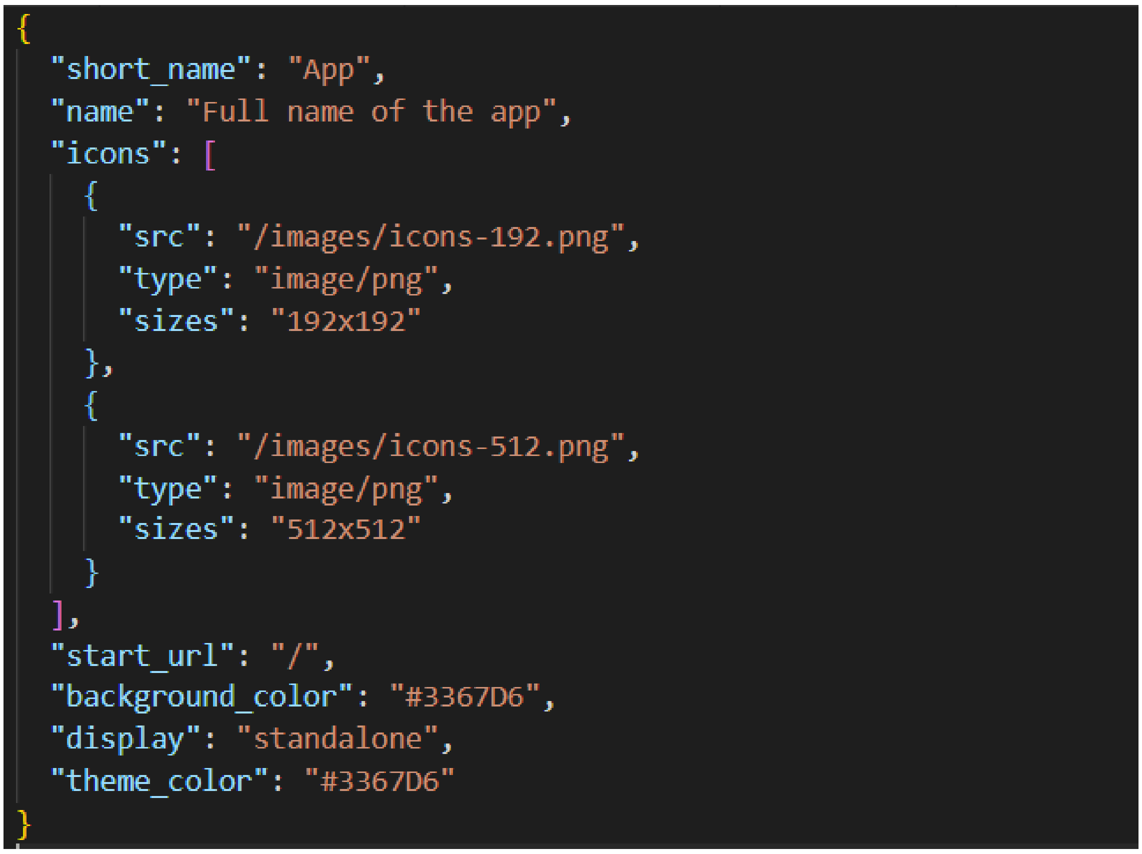
As can be seen from the figure, a promise has to be fulfilled before the event “activate” can be triggered. Firstly, it will check the new cache named “newCacheList” and look for any strange cache names that have not been defined. If any cache found with that condition, it will be deleted. When the promise is completed, then the event “activate” is called.

### 3.2.1.6 Handling responsive images with service worker

During install step, developers can cache the images by install both low and high-res version of the image, or just one version of them to save up storage space. Moreover, images’ style is usually set with fixed width and height to prevent wrong size of image being displayed due to different type of screen in offline mode.

### 3.2.2 Manifest file

Another important part of progressive web application is manifest [18]. Manifest is an JSON file which covers all the information of the web application such as name, author, description, icon etc. It provides those information to the browser and shows browser how to react when the web application is installed to users' mobile devices or desktop. A typical manifest file will be described as in the figure below:



```
{
  "short_name": "App",
  "name": "Full name of the app",
  "icons": [
    {
      "src": "/images/icons-192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/images/icons-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/",
  "background_color": "#3367D6",
  "display": "standalone",
  "theme_color": "#3367D6"
}
```

Figure 6. example of manifest file.

Usually in the manifest file there will be different size of icon listed. In addition, there will be some other information such as start_url, background_color, theme color_color display and so on [Figure 6].

Display is a very important option to note. There are four displays that can be chosen: standalone (the default one of progressive web application), fullscreen, minimal-UI and browser. The main differences between those modes are:

- Standalone: the application layout will look like native app.
- Fullscreen: the application will be displayed fullscreen.
- Minimal-UI: in this mode, it looks almost the same as standalone. Although, there are still a few browser UI applied (depends on what the browser is).
- Browser: the application will be displayed with full browser experience.

Another option that needs to be handled carefully is start_url. It defines what is the start page when user opens the application. Sometimes it is better to lead the customers to other pages other than the landing page.

When the application is launched, it will take awhile to render all the content. For the time being, a splash screen [Figure 7] will be created using background_color and icon that has been defined in the manifest.

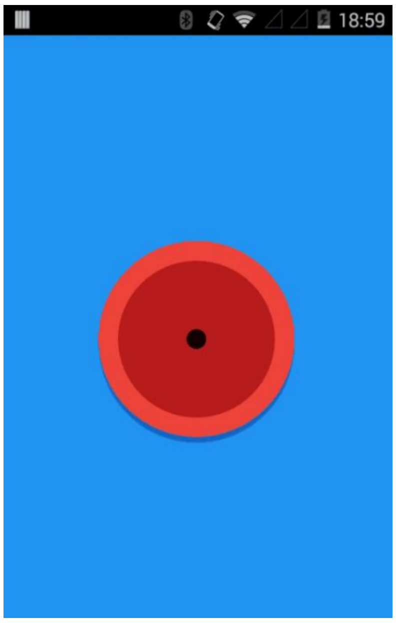


Figure 7. Example of a splash screen.

### 3.2.3 Lighthouse

Using Lighthouse [19], an open-source and automated tool which is developed by Google, is a nice way to test if the application is qualified to be a PWA. It is not only built for PWA testing purpose, but to check a website's overall quality. Lighthouse can be used to check the performance, accessibility and security aspects.

There are several ways to use Lighthouse:

- Using command line.
- Using Chrome DevTools.
- Using Lighthouse from a web UI.

#### 3.2.3.1 Using command line

The prerequisite for this method is simple. Users need to have Google Chrome for Desktop ready at hand. Next step is to download newest version of Node. Once Node is installed, users can install Lighthouse as a global module with Node command:

```
npm install -g lighthouse
```

Then to start an audit, users can use the following command:

```
lighthouse <url>
```

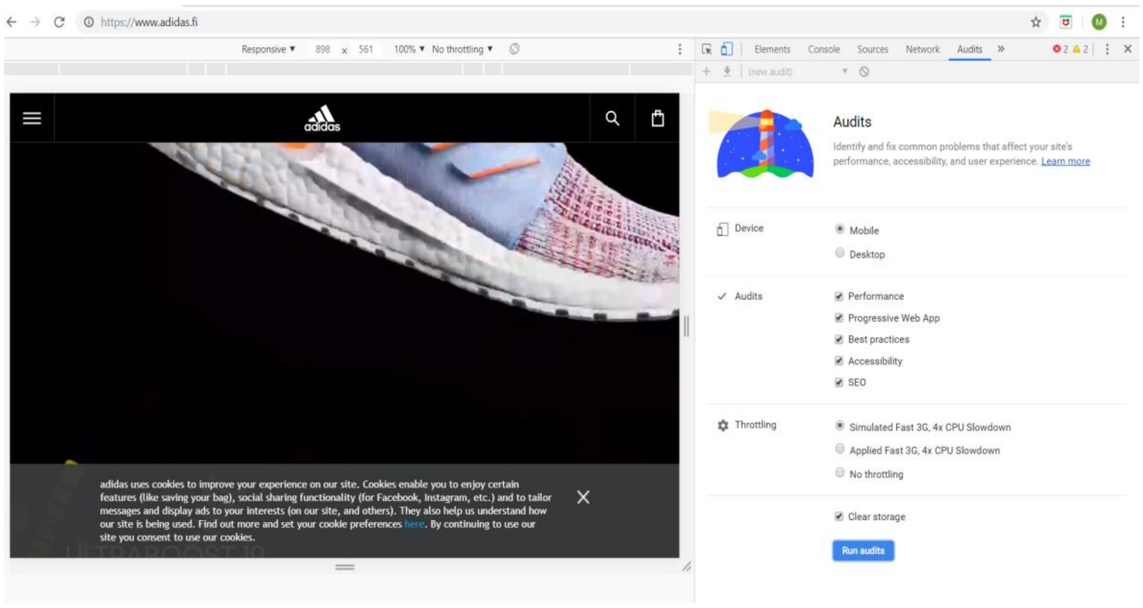
Option of the audit can be found by typing:

```
lighthouse --help
```

#### 3.2.3.2 Using Chrome DevTools

This method has the same perquisite as the previous one: having Google Chrome for Desktop ready at hand. Steps to use Lighthouse are:

- Navigating to the URL that needs to be audited.
- Opening Google DevTools.
- Going to audit tab [Figure 8].



Audits

Identify and fix common problems that affect your site's performance, accessibility, and user experience. Learn more

Device: [✓ Mobile] Desktop

Audits: ✓ Performance, ✓ Progressive Web App, ✓ Best practices, ✓ Accessibility, ✓ SEO

Throttling: [✓ Simulated Fast 3G, 4x CPU Slowdown] Applied Fast 3G, 4x CPU Slowdown No throttling

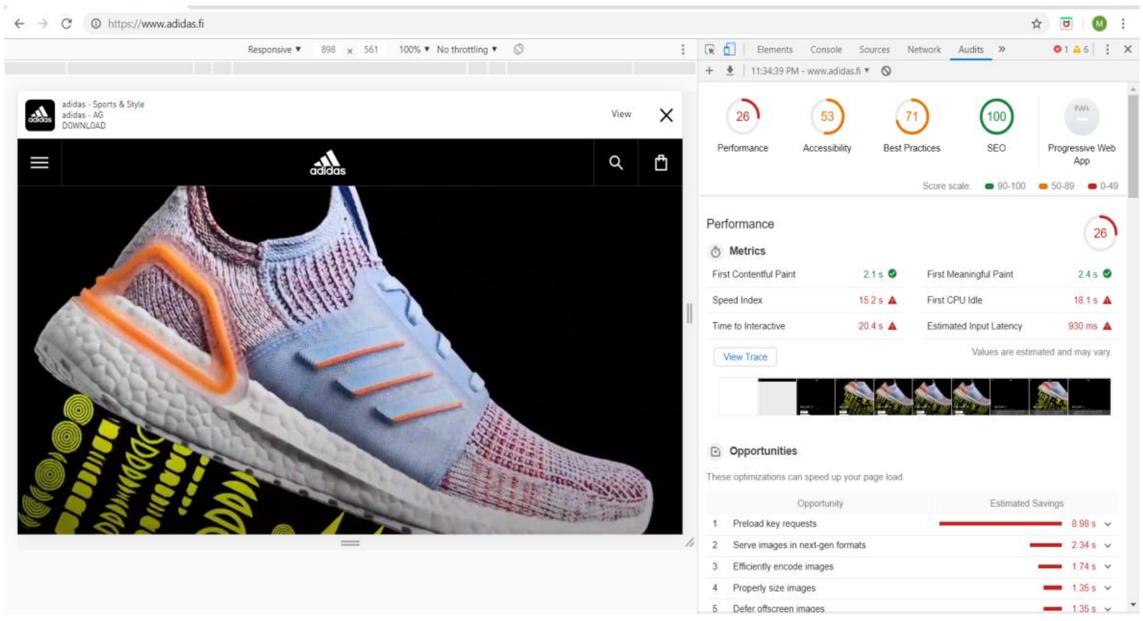
✓ Clear storage

Run audits

adidas uses cookies to improve your experience on our site. Cookies enable you to enjoy certain features (like saving your bag), social sharing functionality (for Facebook, Instagram, etc.) and to tailor messages and display ads to your interests (on our site, and others). They also help us understand how our site is being used. Find out more and set your cookie preferences here. By continuing to use our site you consent to use our cookies.

Figure 8. Audit tab in Google DevTools.

- Clicking “Run audit” to start the process. A result board will be shown [Figure 9].



Performance 26

Metrics

| | | | |
|---|---|---|---|
| First Contentful Paint | 2.1 s | First Meaningful Paint | 2.4 s |
| Speed Index | 15.2 s | First CPU Idle | 18.1 s |
| Time to Interactive | 20.4 s | Estimated Input Latency | 930 ms |

View Trace

Values are estimated and may vary.

Opportunities

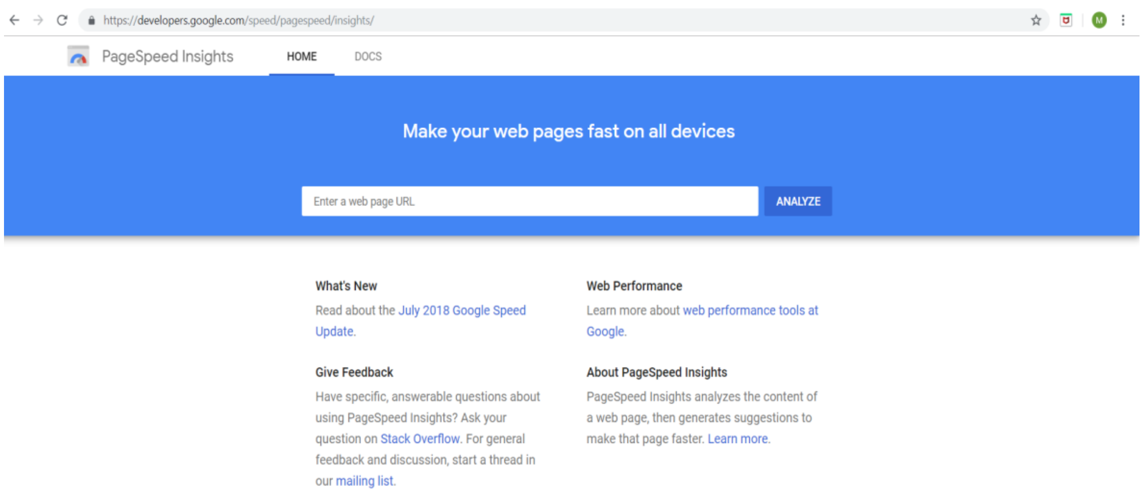
These optimizations can speed up your page load.

| | Opportunity | Estimated Savings |
|---|---|---|
| 1 | Preload key requests | 8.98 s |
| 2 | Serve images in next-gen formats | 2.34 s |
| 3 | Efficiently encode images | 1.74 s |
| 4 | Properly size images | 1.35 s |
| 5 | Defer offscreen images | 1.35 s |

Figure 9. Result board of an audited webpage.

### 3.2.3.3 Using Lighthouse from a web UI

Another way to audit webpage is to navigate to PageSpeed Insights and type in an URL. The result will be shown accordingly [Figure 10].



Make your web pages fast on all devices

What's New

Read about the July 2018 Google Speed Update.

Web Performance

Learn more about web performance tools at Google.

Give Feedback

Have specific, answerable questions about using PageSpeed Insights? Ask your question on Stack Overflow. For general feedback and discussion, start a thread in our mailing list.

About PageSpeed Insights

PageSpeed Insights analyzes the content of a web page, then generates suggestions to make that page faster. Learn more.

Figure 10. Example of using PageSpeed Insights to audit URL.

### 3.2.4 HTTPS

#### 3.2.4.1 What is HTTPS

In the past, when developing web application, developers usually use HTTP. But now as PWA is claimed to be more secure for users, HTTPS is utilized. HTTPS is a more secure version of HTTP. It is used to help a web browser communicate with websites in a safer way, preventing harms which might come from hackers, intrusive internet service providers and so on.Transport layer security is used to encrypted communication protocol in HTTPS [20].

HTTPS is becoming more and more popular. In this past, its mostly used to protect users' password or private data. Nowadays, it is used more widely in all kind of websites, leaving HTTP behind.

#### 3.2.4.2 Differences between HTTP and HTTPS

If HTTP is used, data will be sent as readable text (not encrypted) back and forth in the connection to the web server which has the IP that corresponds with the desired website. Users might end up being redirected to another website without noticing. Sensitive data such as password or other personal information should not be transmitted over the internet using HTTP ever.

On the other hand, HTTPS-secured servers will redirect users to HTTPS webpages. As a process, users' web browser will check those webpages' security certificate which is issued by legitimate authority. By that way, users can ensure that the desired webpage is real, and sensible data is encrypted properly. Although, security certificate might not be correctly issued all the time, but it helps to lessen the risk of leaking data.

Nowadays, new applications' features such as taking pictures, recording, or offering offline experience require users' permission. As a crucial factor in the permission process, HTTPS is used widely to create PWA to protect users' private data and information.

### 3.2.5 Push notification

Push notification is a way for service provider to constantly re-engage with users, even if the application is not opened/active. It is a noticing message that is sent from the server to users, usually to inform an event or an update of the application, or even a simple message to remind users to continue using the application.

Notifications API and Push API are combined to create push notification. Notifications API helps users to receive notifications from the app. On the other hand, Push API helps the developers to handle notification messages from the server using service worker. Both of the APIs use service worker, which are responsible for handling events in the background and display them to the application.

### 3.2.6 Workbox

Workbox [21] is a library which has ready-made practices to ease the process of cache assets. With the help of Workbox, developers will save lots of time when working with service worker.

Workbox can be installed using NPM with the command:

```
npm install workbox-cli -g
```

More information on how to configurate workbox will be mentioned in the next chapter.

# 4 EXAMPLE OF BUILDING PROGRESSIVE WEB APPLICATION

In this chapter, the way to build a basic progressive web application from scratch is discussed. Additional tools that helps to create progressive web application are mentioned along the chapter as well.

## 4.1 Choose suitable code editor

Choosing a suitable code editor is crucial since it will boost the speed of the whole process if a suitable code editor is used. All in all, every code editor has its own strengths and weaknesses, but there are a few editors are favored more by developers such as Visual Studio Code, Atom or Sublime. All of them has good visual user interface with possibility to add themes. Visual Studio Code and Sublime are quite light-weighted than Atom. While most of the code editors nowadays is free, Sublime comes with a fee and free trial version.

In this thesis, Visual Studio Code will be use since it is fast, intelligent, free and git-intergrated. By supporting developers with fast completing shortcut and smart searching with nice UI, Visual Studio Code will help to save lots of time.

## 4.2 Set up file directory

Basically, a new folder should be created to organize the project files [Figure 11]. Inside of that folder, three main files are created: index.html, styles.css and main.js. The HTML file is the skeleton of the application. Next, CSS file is the beauty of the application. Last but not least, JS file will help to make the application more dynamic with its functions.

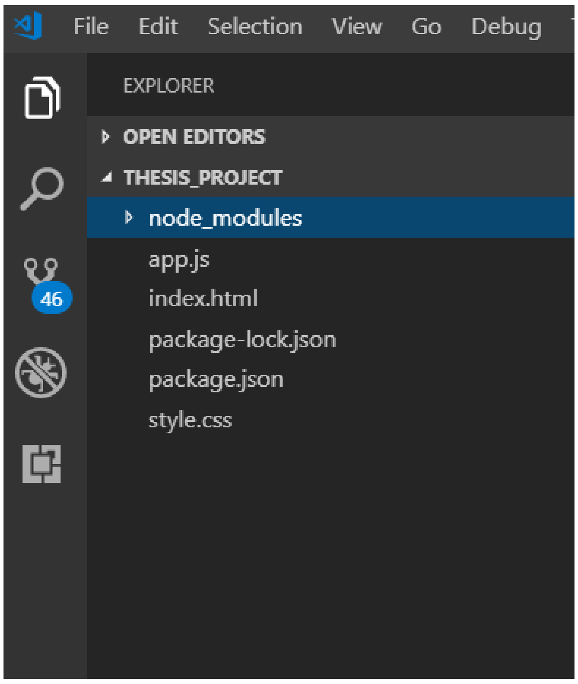


Figure 11. Example of file directory.

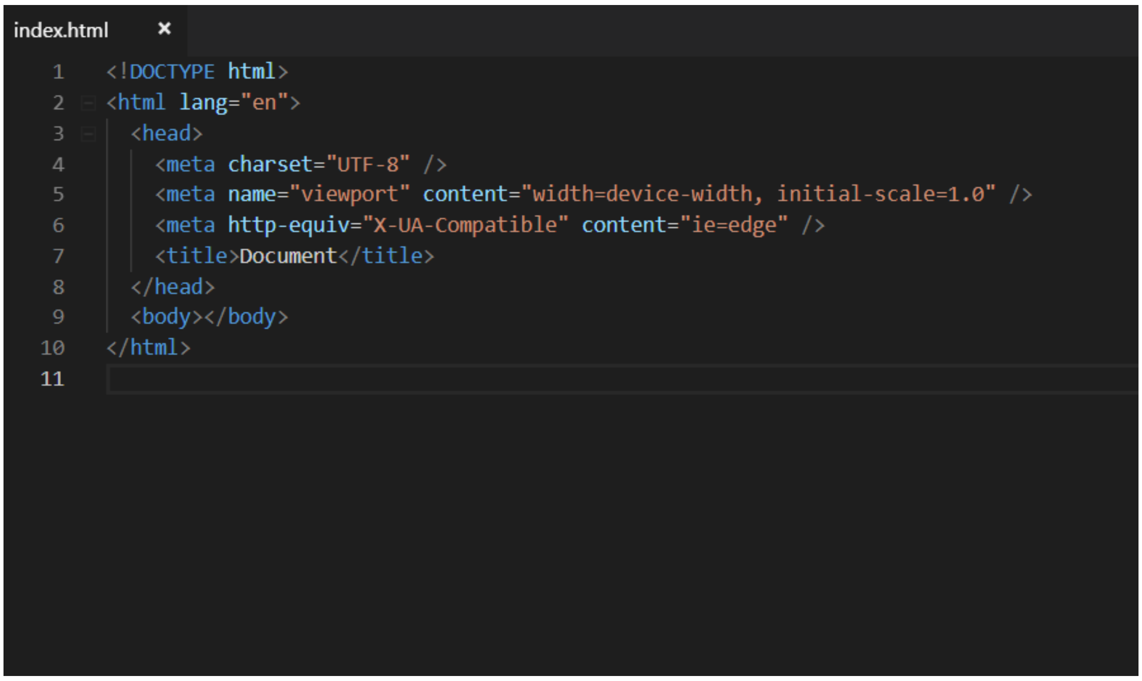
As can be seen from above figure, there are also node_modules folder and package.json file. Those file will be discussed later in this thesis.

## 4.3 Build a web application

Building a web application is the next step, since progressive web application is just an upgraded version of it using the help of service worker and manifest. In offline mode, it should still be able to show some meaningful news from the last cache session.

As every normal web application, the first step is to create a solid structure with HTML [Figure 12]. A common empty template could be created quickly with:

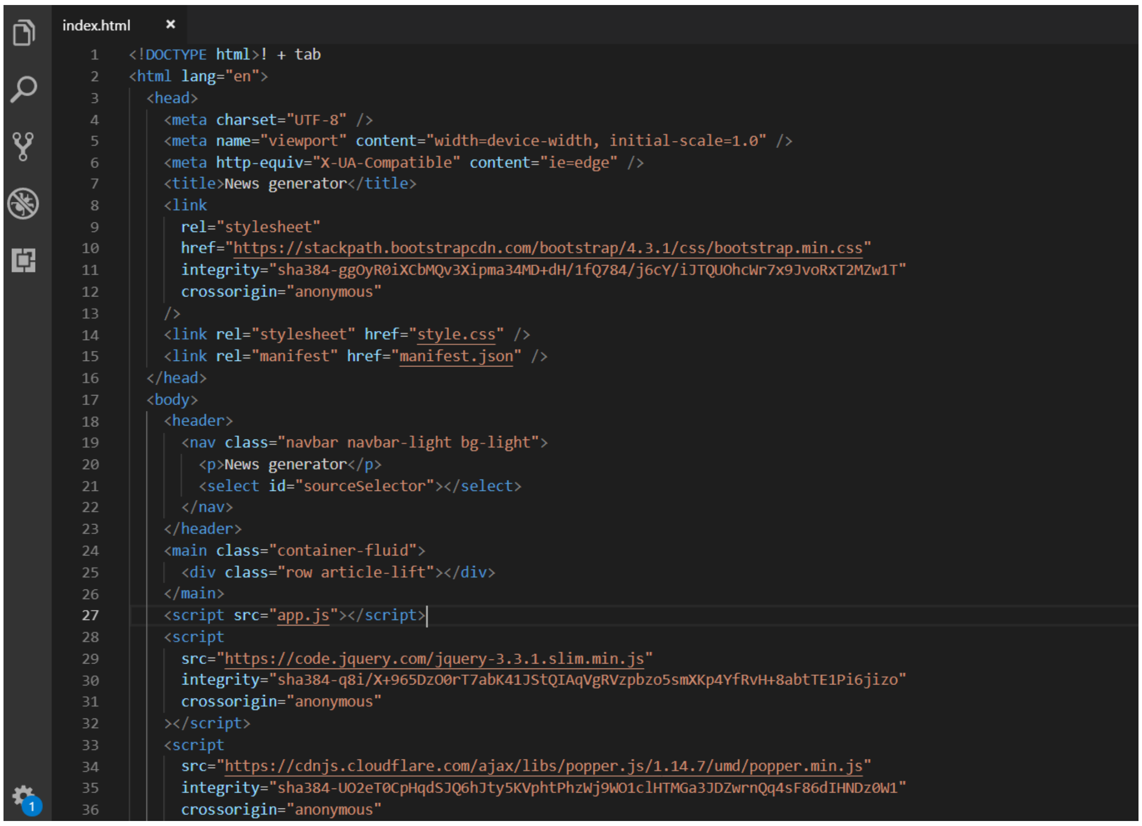
```
! + tab
```


index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Document</title>
  </head>
  <body></body>
</html>

```

Figure 12. First form of template quickly built with shortcut.

The next step is to create a completed template [Figure 13]. As can be seen from the figure below, Bootstrap is used, so the structure is quite simple. Boostrap is a CSS library which has been around for years. It supports developers with ready-to-use components and styles. Script of Pooper.js and Bootstrap.js need to be added at the end of the file.

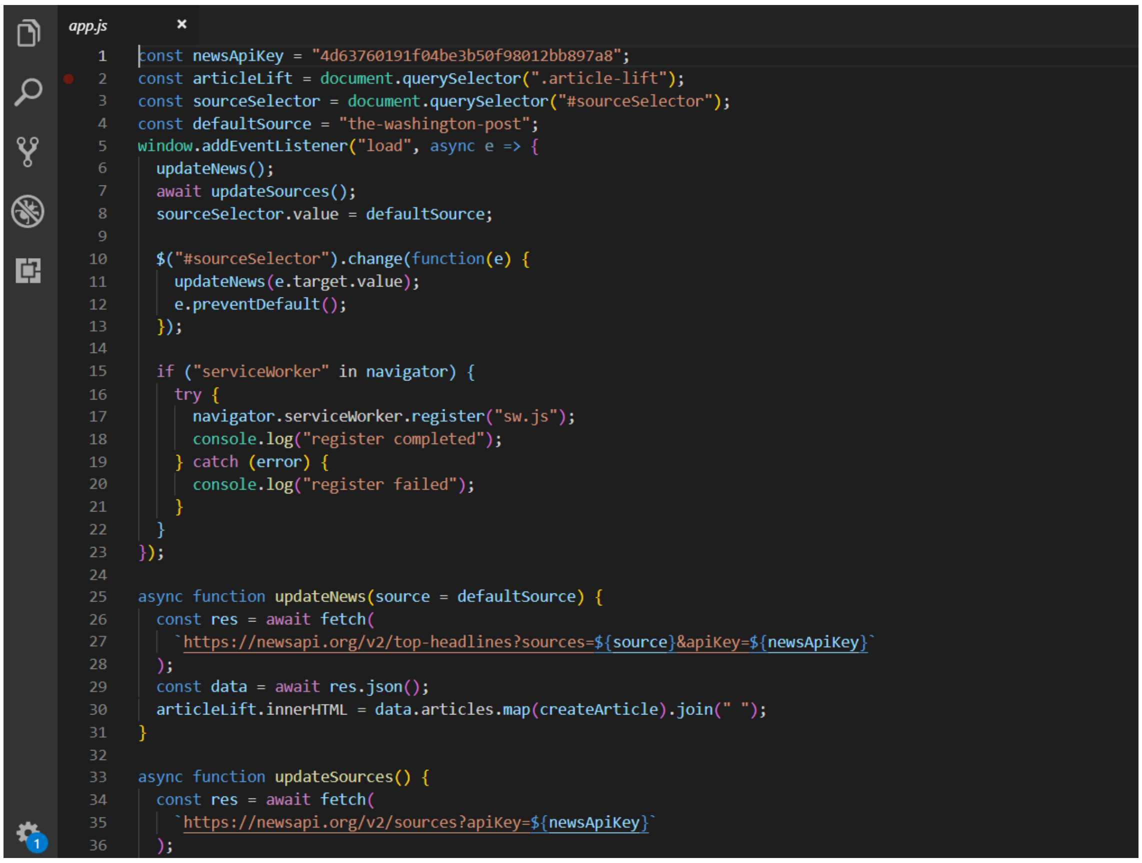


```html
<!DOCTYPE html>! + tab
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>News generator</title>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous"
    />
    <link rel="stylesheet" href="style.css" />
    <link rel="manifest" href="manifest.json" />
  </head>
  <body>
    <header>
      <nav class="navbar navbar-light bg-light">
        <p>News generator</p>
        <select id="sourceSelector"></select>
      </nav>
    </header>
    <main class="container-fluid">
      <div class="row article-lift"></div>
    </main>
    <script src="app.js"></script>
    <script
      src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
      integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
      crossorigin="anonymous"
```

Figure 13. Completed structure of the application.

With the help of Bootstrap, most of the styles is done by adding names of pre-made styles to the elements. But if customize styles are desired, then those styles could be added to styles.css file to override Bootstrap's styles.

The next file is main.js [Figure 14]. In this file, there are functions and events which will help the application run smoothly and handle the API well. Moreover, the service worker will also be registered in this file.



```javascript
const newsApiKey = "4d63760191f04be3b50f98012bb897a8";
const articleLift = document.querySelector(".article-lift");
const sourceSelector = document.querySelector("#sourceSelector");
const defaultSource = "the-washington-post";
window.addEventListener("load", async e => {
  updateNews();
  await updateSources();
  sourceSelector.value = defaultSource;

  $("#sourceSelector").change(function(e) {
    updateNews(e.target.value);
    e.preventDefault();
  });

  if ("serviceWorker" in navigator) {
    try {
      navigator.serviceWorker.register("sw.js");
      console.log("register completed");
    } catch (error) {
      console.log("register failed");
    }
  }
});

async function updateNews(source = defaultSource) {
  const res = await fetch(
    `https://newsapi.org/v2/top-headlines?sources=${source}&apiKey=${newsApiKey}`
  );
  const data = await res.json();
  articleLift.innerHTML = data.articles.map(createArticle).join(" ");
}

async function updateSources() {
  const res = await fetch(
    `https://newsapi.org/v2/sources?apiKey=${newsApiKey}`
  );
```

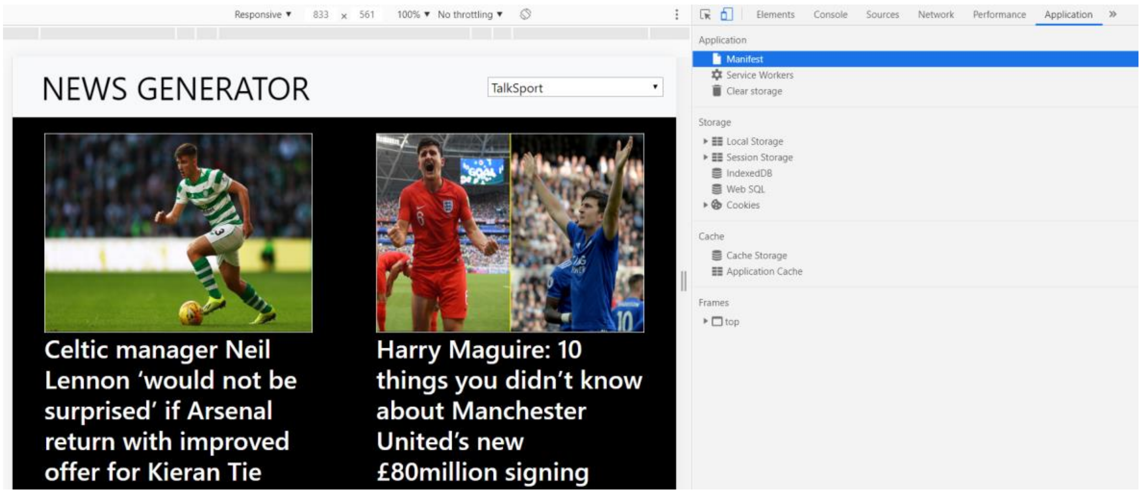
Figure 14. Service worker registration in JavaScript file.

## 4.4 Upgrade to progressive web application

In the previous section 4.3, a basic web application is built. The next step is to upgrade it to progressive web application.

### 4.4.1 Generate manifest file

As can be seen from the figure below, manifest in the first item in the list of application's tab [Figure 15]. Below manifest is service worker tab, which will be discussed later in 4.4.2.

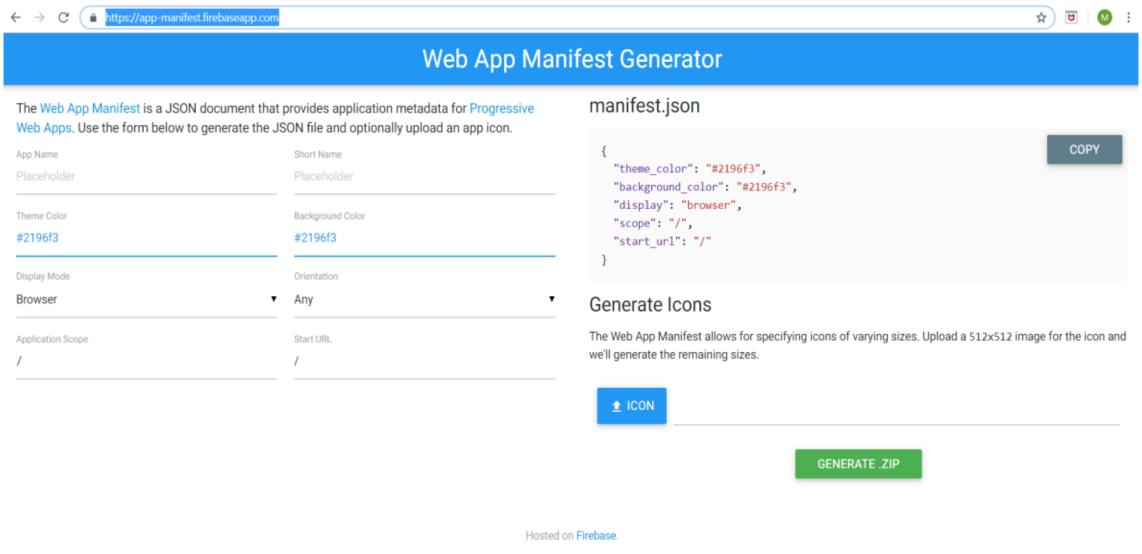


Celtic manager Neil Lennon 'would not be surprised' if Arsenal return with improved offer for Kieran Tie

Harry Maguire: 10 things you didn't know about Manchester United's new £80million signing

Figure 15. Application tab in Google DevTools.

Manifest file can be manually filled in form of a JSON file. Or it can also be generated using some generators online [Figure 16]. One example is https://app-manifest.firebaseapp.com/. The user interface will be demonstrated in the figure below:


https://app-manifest.firebaseapp.com
Web App Manifest Generator

The Web App Manifest is a JSON document that provides application metadata for Progressive Web Apps. Use the form below to generate the JSON file and optionally upload an app icon.

App Name: Placeholder
Short Name: Placeholder
Theme Color: #2196f3
Background Color: #2196f3
Display Mode: Browser
Orientation: Any
Application Scope: /
Start URL: /

manifest.json

```
{
  "theme_color": "#2196f3",
  "background_color": "#2196f3",
  "display": "browser",
  "scope": "/",
  "start_url": "/"
}
```

Generate Icons

The Web App Manifest allows for specifying icons of varying sizes. Upload a 512x512 image for the icon and we'll generate the remaining sizes.

Figure 16. Example of manifest generator.

Once all the information is filled, it will create a manifest file which need to be put in the project folder.

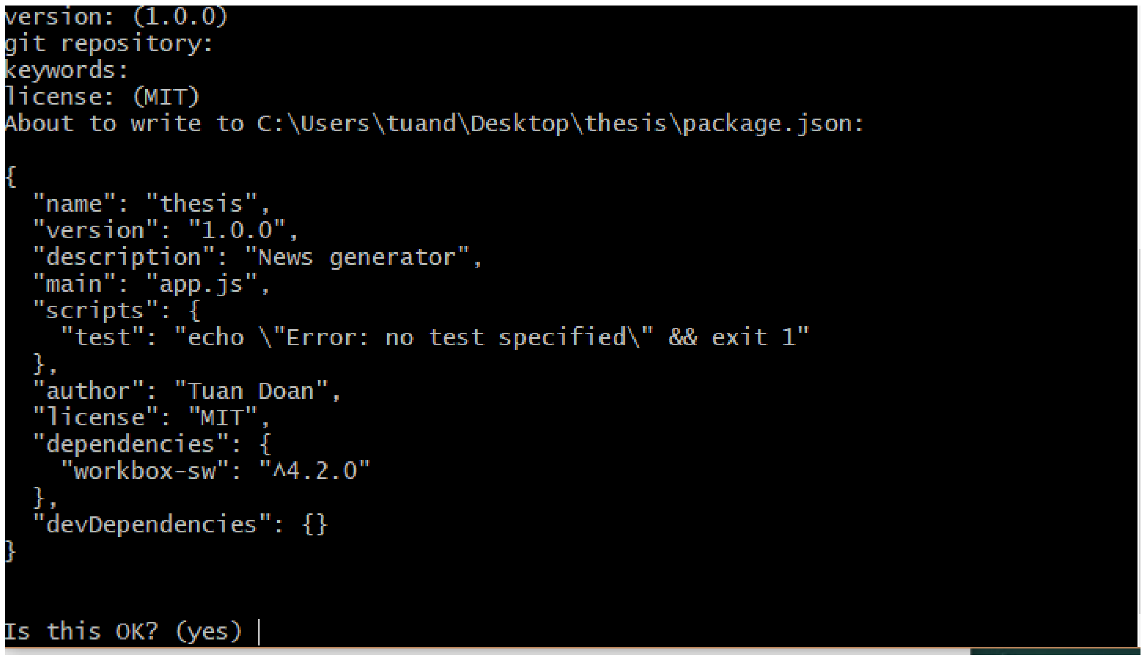
### 4.4.2 Adding service worker

Service worker is simply a plain JavaScript file that listens to events and performs some certain acts based on those events. Workbox will be used to help maintaining the service worker file better.

First thing that need to be done is turning the project to an NPM project. NPM stands for Node Package Manager, which is usually use to handle dependencies for projects. Newest version of Node need to be installed to use the command. After finishing the installation, the project can be turned NPM project with simple command:

```
npm init
```

There will be a few setting that need to be filled after that [Figure 17].



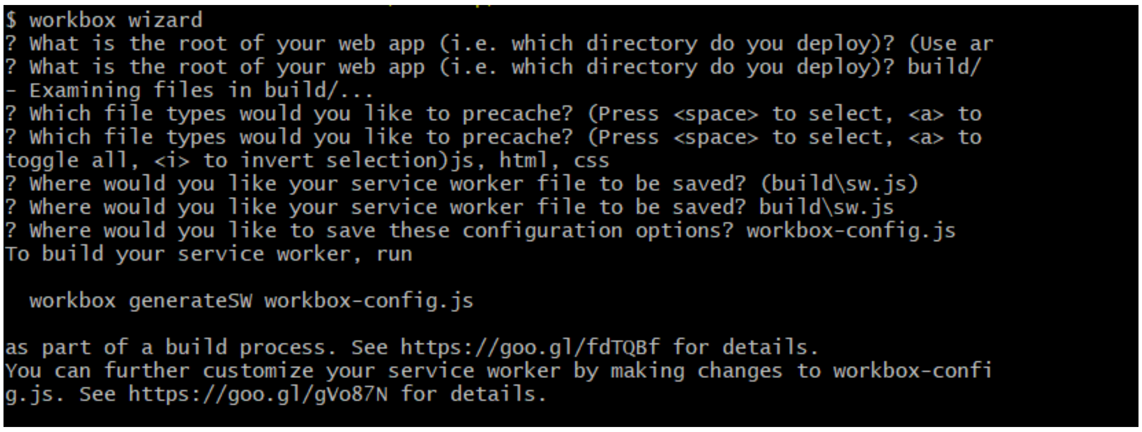
```
version: (1.0.0)
git repository:
keywords:
license: (MIT)
About to write to C:\Users\tuand\Desktop\thesis\package.json:

{
  "name": "thesis",
  "version": "1.0.0",
  "description": "News generator",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Tuan Doan",
  "license": "MIT",
  "dependencies": {
    "workbox-sw": "^4.2.0"
  },
  "devDependencies": {}
}


Is this OK? (yes) |
```

Figure 17.Example of NPM init command.

After Workbox is installed as instructed in previous chapter, it is time to configurate the option. Firstly, this command will be used:

```
workbox wizard
```

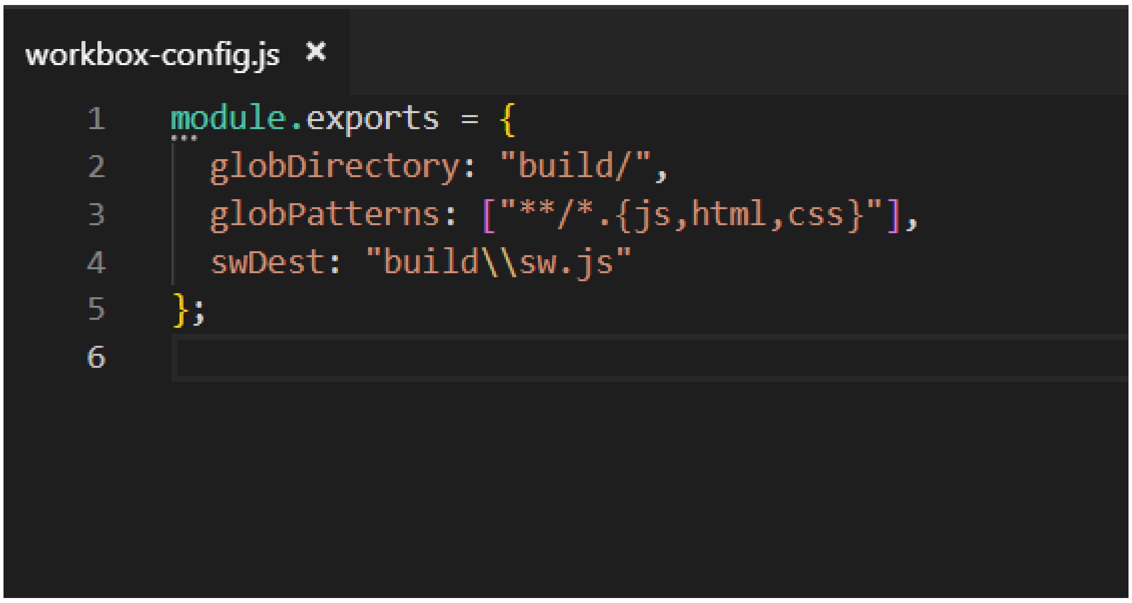
After that, there will be options to choose directory to deploy, and which files to precache [Figure 18]. This is example when all the questions are answered:



```
$ workbox wizard
? What is the root of your web app (i.e. which directory do you deploy)? (Use ar
? What is the root of your web app (i.e. which directory do you deploy)? build/
- Examining files in build/...
? Which file types would you like to precache? (Press <space> to select, <a> to
? Which file types would you like to precache? (Press <space> to select, <a> to
toggle all, <i> to invert selection)js, html, css
? Where would you like your service worker file to be saved? (build\sw.js)
? Where would you like your service worker file to be saved? build\sw.js
? Where would you like to save these configuration options? workbox-config.js
To build your service worker, run

  workbox generateSW workbox-config.js

as part of a build process. See https://goo.gl/fdTQBf for details.
You can further customize your service worker by making changes to workbox-confi
g.js. See https://goo.gl/gVo87N for details.
```

Figure 18. Example of using workbox wizard.

And what it does is to generate a file named “workbox-config.js” in the root folder:



```
workbox-config.js ×

module.exports = {
  globDirectory: "build/",
  globPatterns: ["**/*.{js,html,css}"],
  swDest: "build\\sw.js"
};

```

Figure 19.Auto-generated workbox config file.

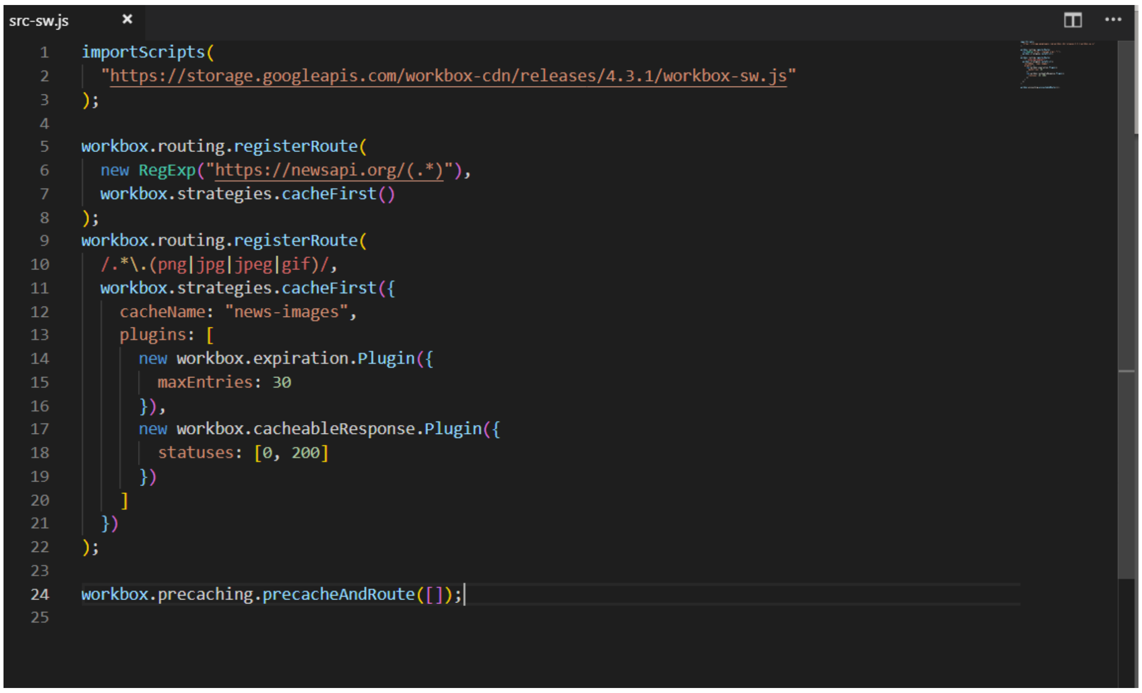
Then, an additional file need to be created in the root folder and named “src-sw.js”. Next thing is to add that url to the module in the workbox-config.js file:

```
swSrc: "src-sw.js"
```

From this moment, any change that applied in the file src-sw.js will be injected to sw.js by using the command:

```
workbox injectManifest
```

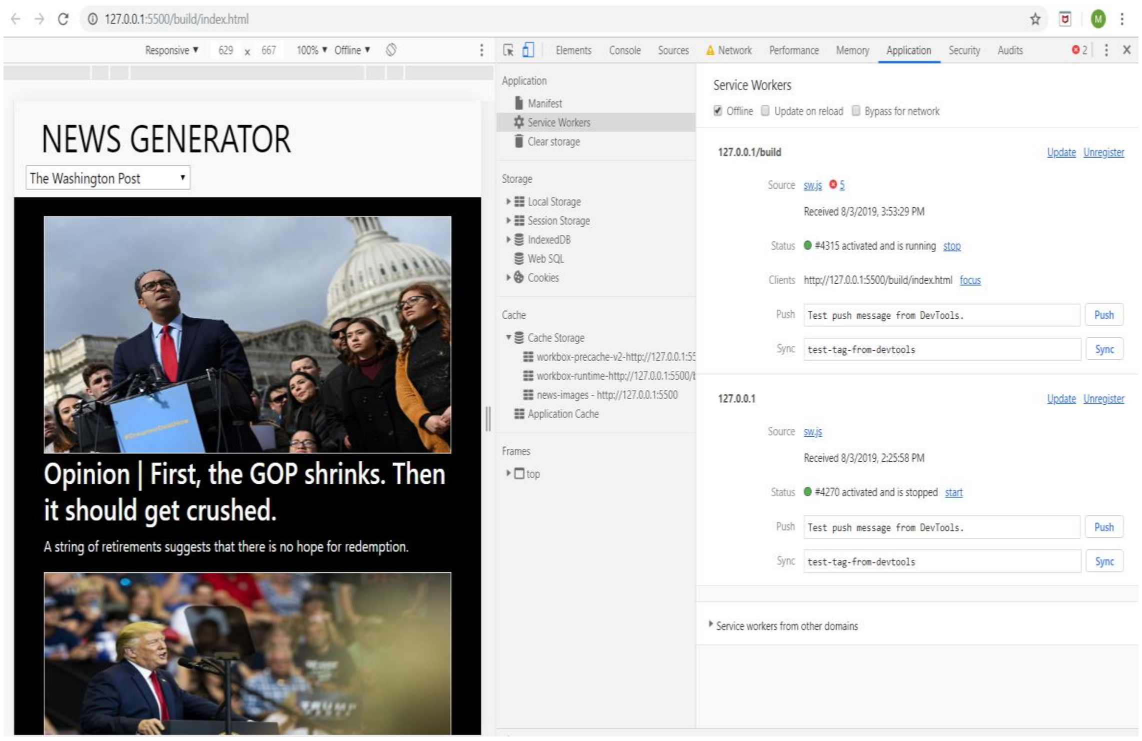
After import script url and adding configuration, src-sw.js will look like the figure below:



```
importScripts(
  "https://storage.googleapis.com/workbox-cdn/releases/4.3.1/workbox-sw.js"
);

workbox.routing.registerRoute(
  new RegExp("https://newsapi.org/(.*)"),
  workbox.strategies.cacheFirst()
);
workbox.routing.registerRoute(
  /.*\.(png|jpg|jpeg|gif)/,
  workbox.strategies.cacheFirst({
    cacheName: "news-images",
    plugins: [
      new workbox.expiration.Plugin({
        maxEntries: 30
      }),
      new workbox.cacheableResponse.Plugin({
        statuses: [0, 200]
      })
    ]
  })
);

workbox.precaching.precacheAndRoute([]);
```

Figure 20. Example of service worker configuration.

As can be seen from the figure, maximum thirty images will be cached, which need to have status from 0 to 200. And the routing is set to the webpage which offers free API. After that, the webpage need to be refreshed to cache the new content. Then, even in offline mode, the webpage will still able to give meaningful contents [Figure 21].


127.0.0.1:5500/build/index.html
Responsive ▾ 629 x 667 100% ▾ Offline ▾
Elements Console Sources Network Performance Memory Application Security Audits
NEWS GENERATOR
The Washington Post
Opinion | First, the GOP shrinks. Then it should get crushed.
A string of retirements suggests that there is no hope for redemption.
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
workbox-precache-v2-http://127.0.0.1:55
workbox-runtime-http://127.0.0.1:5500/t
news-images - http://127.0.0.1:5500
Application Cache
Frames
top

Service Workers
☑ Offline ☐ Update on reload ☐ Bypass for network

127.0.0.1/build — Update Unregister
Source: sw.js 5
Received 8/3/2019, 3:53:29 PM
Status: #4315 activated and is running stop
Clients: http://127.0.0.1:5500/build/index.html focus
Push: Test push message from DevTools. Push
Sync: test-tag-from-devtools Sync

127.0.0.1 — Update Unregister
Source: sw.js
Received 8/3/2019, 2:25:58 PM
Status: #4270 activated and is stopped start
Push: Test push message from DevTools. Push
Sync: test-tag-from-devtools Sync

Service workers from other domains

Figure 21.Example of PWA running in offline mode.

As can be seen from the figure above, the webpage is in offline mode. It is now capable of loading meaningful contents such as images, title, and the source. Without the help of service worker and workbox, the webpage will just display an “offline dinosaur”. In the example, the webpage is run on local server. If it is uploaded to any web hosting sources, the result will also remain the same.

# 5 CONCLUSION

This thesis's purpose was to offer a glance at the history of application development and the rising power of progressive web application. Web, mobile, hybrid and progressive web applications have been introduced and discussed clearly throughout the thesis. The advantages and disadvantages of every type have been listed to provide a better idea of which one is gaining traction in the industry lately.

It is not surprising that progressive web application is becoming more and more popular because of its reliability and performance. Its strengths are discussed carefully during the thesis to emphasize the ability of combining both web application and native application. Moreover, all of its core technologies are also mentioned with details and examples.

The approach to build a progressive web application has been demonstrated in chapters 3-4 of the thesis to offer a good start for those who want to learn more about trending technologies.

# REFERENCES

[1] buildfire.com (2019) [online] available at: https://buildfire.com/app-statistics/ [Accessed 28 Sep. 2019].

[2] java.com (2019) [online] available at: https://www.java.com/en/ [Accessed 28 Sep. 2019].

[3] tutorialspoint.com (2019) [online] available at: https://www.tutorialspoint.com/csharp/index.htm [Accessed 28 Sep. 2019].

[4] developer.apple.com (2019) [online] available at: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html [Accessed 28 Sep. 2019].

[5] swift.com (2019) [online] available at: https://www.swift.com/ [Accessed 28 Sep. 2019].

[6] cplusplus.com (2019) [online] available at: http://www.cplusplus.com/ [Accessed 28 Sep. 2019].

[7] python.org (2019) [online] available at: https://www.python.org/ [Accessed 28 Sep. 2019].

[8] kotlin.com (2019) [online] available at: https://kotlinlang.org/ [Accessed 28 Sep. 2019].

[9] realm.io (2019) Report [online] available at: https://realm.io/realm-report/ [Accessed 24 Apr. 2019].

[10] steelkiwi.com [online] available at: https://steelkiwi.com/blog/how-to-publish-your-app-on-the-app-store-and-google-play/ [Accessed 15 May. 2019].

[11] javascript.com [online] available at: https://www.javascript.com/ [Accessed 16 May. 2019].

[12] nodejs.org [online] available at: https://nodejs.org/en/ [Accessed 16 May. 2019].

[13] wikipedia.org (2019) Report [online] available at: https://en.wikipedia.org/wiki/Web_application [Accessed 16 May. 2019].

[14] reactjs.org (2019) [online] available at: https://reactjs.org/ [Accessed 28 Sep. 2019].

[15] vuejs.org.org (2019) [online] available at: https://vuejs.org/ [Accessed 28 Sep. 2019].

[16] angular.io (2019) [online] available at: https://angular.io/ [Accessed 28 Sep. 2019].

[17] Google Developers. (2019) Service worker [online] available at: https://developers.google.com/web/fundamentals/primers/service-workers/ [Accessed 19 May. 2019].

[18] Google Developers. (2019) Manifest [online] available at: https://developers.google.com/web/fundamentals/web-app-manifest/ [Accessed 23 May. 2019].

[19] Google Developers. (2019) Lighthouse [online] available at: https://developers.google.com/web/tools/lighthouse/ [Accessed 26 May. 2019].

[20] Wikipedia.org (2019) HTPPS [online] available at: https://en.wikipedia.org/wiki/HTTPS

[Accessed 28 May. 2019].

[21] Google Developers. (2019) Workbox | precaching [online]  available at: https://developers.google.com/web/tools/workbox/modules/workbox-precaching

[Accessed 10 Jun. 2019].