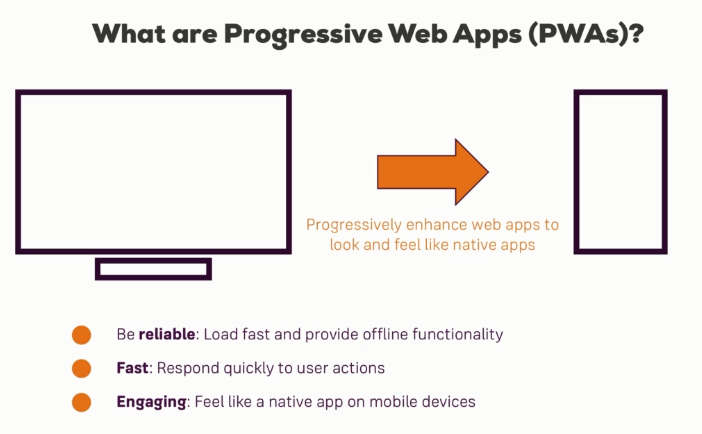
PWA  
(Progressive Web Apps)

# Getting Started

## What are Progressive Web Apps?

Docs: <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps>

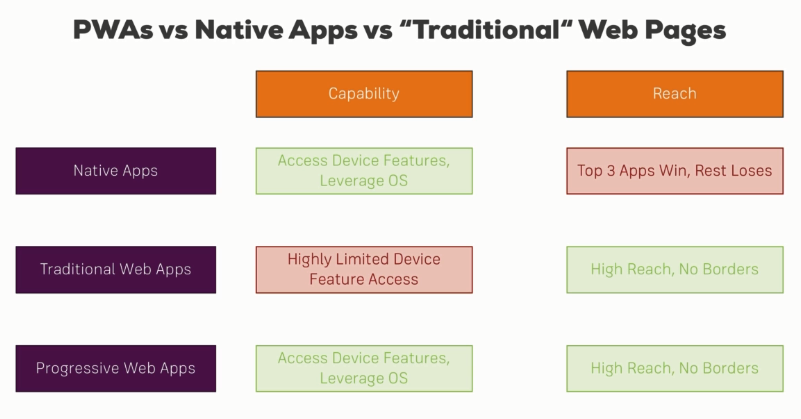


* Progressive web apps in the end describe a set of features of technologies you can add to your existing web applications to turn them into native apps like native mobile app like experiences using device features like the camera and providing offline access.
* You **progressively enhance** your existing web pages to feel and work more like native mobile apps.
* Things like your app working if you're offline, it's having an icon on the home screen, things like accessing the device camera or the location, synchronizing data in the background. These are all features which were hard to do in web applications in the past, nowadays we got browser support in a lot of browsers for that and we can therefore use that.
* You progressively enhance a web application. It's not an all or nothing move. It's not like it doesn't work on older browsers anymore. It just means if you have a modern browser on a mobile device, you get an awesome experience otherwise you'll get the experience you've gotten anyways.

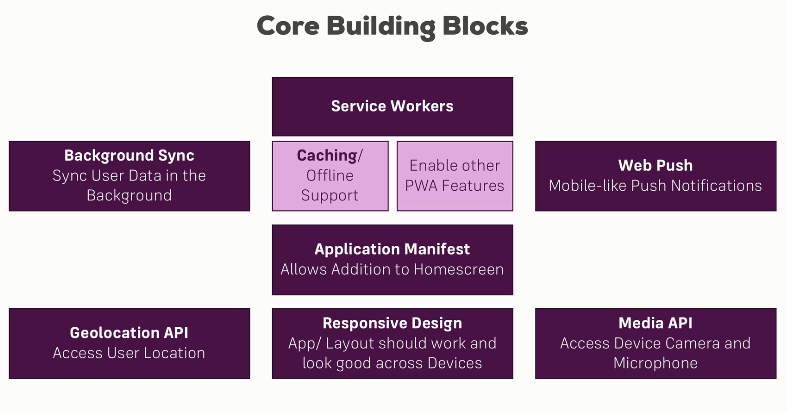
### Three main things that summarizes PWAs

1. They should be **reliable** which means they should **load fast** and even work if you're **offline**, at least a part of the application should work if you're offline.
   1. Now this is really a core thing of progressive web apps. We're talking a lot about that initial load, the first time you visit an application when you open it because if you consider an application running on your mobile device, most of them start up pretty fast, so long loading times you don't want that.
2. Additionally it should be **fast**, not just during load up but also once it runs, it should react to user input, it should provide animations, it should be able to access native device features in an intuitive way
3. And we also want to make sure that it's **engaging**. We want to get our users back into the app, we want to offer features like push notifications to inform them even if the application is closed.

## PWAs Vs Native Mobile Apps



## PWA Core Building Blocks



### Service Workers

* **Service workers are basically JavaScript running in a background process, even if your application is closed.**
* Service workers are supported in modern browsers like Chrome.
* Service workers are a core building block because they allow us to get offline access, to cache some files and serve them if we don't have internet connection.
* And they also give us access to other progressive web app related features for example background synchronization, sending a request once internet connection is re-established. Push notifications would be another example because they are running in the background independent of currently opened tabs.

### Application Manifest

* The Application Manifest makes your application **installable** on home screens, not installed through App Store but instead, you can basically install a web app.

### Responsive Design

* App should work and look good across devices.

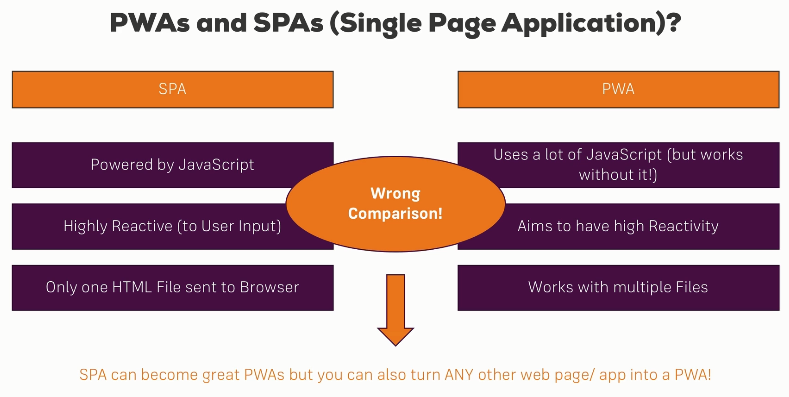
### Geolocation API

* To access user’s location.

### Media API

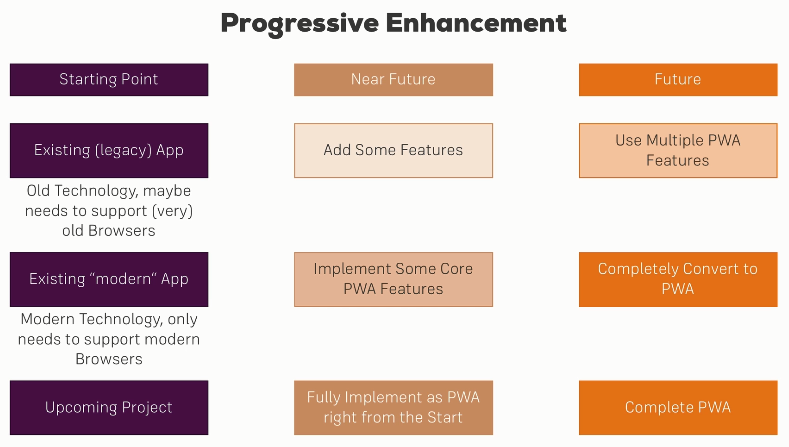
* To access device camera and microphone.

## Comparing PWAs with SPAs



* You can turn any single page application into a progressive web app and the same is true for any traditional multi-page application where you render your views from your server.
* A progressive web app is just a collection of technologies, can be used anywhere.

## What is “Progressive Enhancement”



* The term “Progressive” in PWA basically means that we can progressively enhance our web application.
* With existing project, we can add features step-by-step and you can stop at any given point. You can stop after adding an app manifest and a basic service worker. You can stop after you implement the basic caching, advanced caching, push notifications. You can stop all the time.
* For the upcoming project, you can start from scratch obviously. So you can fully implement it as a progressive web app right from the start and plan everything with that in mind.
* It’s not all or nothing. You don't have to use 0 or 100%, you can absolutely use 40% of all the features, pick the ones which enhance your application and add them.
* So basically you can “progressively” enhance any application (legacy, modern, upcoming) to use different features of PWA.

# Understanding the App Manifest

## App Manifest File

* The web app manifest is just one single file you add to your project, where you can pass some extra information to the browser.
* Now the browser can take this information to display some things about your application differently and the browser depending on which operating system it's running on can even allow your app to install itself on the home screen of your user. Then it will feel and look like a native app, so we can open it like we open any native app.

### Advantages –

* If users just visit our web page via browser, they might enjoy it but they constantly have to type the URL or manage the bookmark.
* But with a home screen icon which we can get via the app manifest, we increase user interaction with our application.

## Adding the Manifest file

* Web manifest simply is a file we add to our root web folder, so the folder containing our index.html file.
* The web manifest file name must be **manifest.json**
* Now include this manifest.json file in EVERY page (.html file) in your application.

E.g.

<link rel="manifest" href="/manifest.json">

* If you have SPA, you just need to one this in the one index.html file. For multipage applications, you must add this entry in each .html page. This is because you want to ensure that the browser is able to load that manifest with your additional configuration, no matter which URL you visit.

## Understanding App Manifest properties

* MDN Web app manifests   
  <https://developer.mozilla.org/en-US/docs/Web/Manifest>
* Web App manifest explanation by Google.  
  <https://web.dev/add-manifest/>
* **name** – allows us to assign a name to our progressive Web application. The name will be fetched in use by the browser in various places. For example on a splash screen we get once we edit the app to the home screen.
* **short\_name** – This is basically the shortened words of our full name and it is the name which is shown below the app icon. The browser will use this name whenever the space for the name is limited.
* **start\_url** – means which page should get loaded on startup when we tap this app icon.
* **scope** – This means which pages are included in our “PWA Experience”. A dot (.) means all the pages in our web folder.
* **display** – allows us to define how the app should actually load once it was added to the home screen.
  + standalone – means it looks like a full app. And we don't see the browser controls at the top, would still have a toolbar. We don't see URL bar and so on.
  + fullscreen – which basically means that it will cover up the whole screen and whilst standalone still would have a toolbar and basically behave like a standalone app, fullscreen is like the fullscreen native apps you know where you don't see any controls at all. Not typically the choice you take for web applications but you could do that.
  + minimal-ui – this will also feel like a standalone application but with a reduced amount of UI elements.
  + browser – which will open it like a normal web page in the browser.
* **background\_color** –
  + The color will display at the back of your application while it is loading and on the splash screen, so on the loading screen.
  + This is simply something which is only shown a fraction of the time but it allows you to make sure that you control this short period of time.
* **theme\_color** – This controls how for example your toolbar at the top will look like especially on a native device in the tasks switcher.
* **description** – is used for example you use when you save the application as a favorite in the browser. So whatever the browser wants additional description to use it will fall back to this field here.
* **dir** – read direction of your app. Be default it is ltr (left to right).
* **lang** – Default or main language of your application. E.g. en-US. This is also useful for the browser so that it can identify how to load your app and for whom your application actually is.
* **orientation** –
  + here you can basically control it in which orientation your web app should be opened if you tap the homescreen icon.
  + This allows you to basically ensure that what users are viewing the app the way you want to view them.
  + But keep in mind restricting the orientation is something you want to use with care because typically users don't like being restricted there.
  + Allowed values – any, portrait, landscape, portrait-primary.
* **icons** – allows us to configure a set of icons where the browser will choose the best fitting one so best fitting for it to give them the why is device density or screen density and so on. You typically provide same icons with multiple sizes and then the browser chooses the best fit for given device. At least have 48 X 48 size. You can also have 128 X 128 or all the way up to 512 X 512.
* **related\_applications** –
  + here you can basically set up related native applications, not web applications the user might be interested in installing for your app.
  + If you have for example a web app where you all have a real native app as an alternative you might specified here and then the browser can choose if he wants to display this choice to the user.

## Preparing the Emulated Device for ALL PWA Features

* We can set up an emulated Android device to test the PWA.
* Here are some more details about the process and how to prepare the device for the rest of the course though.

### **Setting up Android Studio:**

* In case you don't have Android Studio installed, make sure to do so. It's free!
* You can download it from this URL: <https://developer.android.com/studio/index.html>
* We only install it to get easy access to the Android Virtual Device (AVD) Manager though. You can access that Manager under "Tools" => "Android" => "AVD Manager".
* Detailed instructions on how to create a device with it can be found here: <https://developer.android.com/studio/run/managing-avds.html>

### **Updating Chrome on the Virtual Device**

* With an emulated device up and running, you're well-prepared to test your manifest.json file. For other features you learn about in this course, the pre-installed Chrome browser is too old though (at least at the point of time this course is created).
* You can easily update Chrome on your virtual device though. Get an updated APK (basically the app installation file) from this link: <https://www.apkmirror.com/apk/google-inc/chrome/#variants>
* Feel free to choose the latest one, download and install it. Give the device the permission to install from "unsafe" sources. If it fails, try a different APK version.

## Criteria for installing the Web app (as PWA)

* What does it take to be installable?  
  <https://web.dev/install-criteria/>
* Customizing the install experience: How to provide your own in-app install experience?  
  <https://web.dev/customize-install/>
* Patterns for promoting PWA installation  
  <https://web.dev/promote-install/>

## Adding Browser Specific Properties

* Chrome detects the manifest.json and accordingly will do the needful (e.g. showing popup for ‘Add the Home screen’, etc.)
* Safari does not detect manifest.json as of now. For that, we need to add additional data.

### For Safari

* Add below lines in all the html files in your app.

<!-- For Safari browser -->

  <!-- we want to treat this as a mobile web app which also allows us to add it the home screen -->

  <meta name="apple-mobile-web-app-capable" content="yes">

  <!-- how the status bar will display on the apple devices. -->

  <meta name="apple-mobile-web-app-status-bar-style" content="black">

  <!-- title for your installed web app. By default, it will use title from <title> tag above. -->

  <meta name="apple-mobile-web-app-title" content="PWAGram">

  <!-- which icons to use on Apple devices. BTW apple-icons\*.png are icons optimized for apple. -->

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-57x57.png" sizes="57x57">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-60x60.png" sizes="60x60">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-72x72.png" sizes="72x72">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-76x76.png" sizes="76x76">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-114x114.png" sizes="114x114">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-120x120.png" sizes="120x120">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-144x144.png" sizes="144x144">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-152x152.png" sizes="152x152">

  <link rel="apple-touch-icon" href="/src/images/icons/apple-icon-180x180.png" sizes="180x180">

### For Internet Explorer

* Add below lines in all the html files in your app.

<!-- For Internet Explorer browser -->

  <!-- this will be the image used on this Windows tile then if we save this to the home screen -->

  <meta name="msapplication-TileImage" content="/src/images/icons/app-icon-144x144.png">

  <!-- the background color behind the icon -->

  <meta name="msapplication-TileColor" content="#fff">

### For all browsers

* Add below lines in all the html files in your app.

<!-- This is same as theme\_color property in the manifest.json file. Will be used to other browers -->

  <meta name="theme-color" content="#3f51b5">

# Tips and Tricks

* MDN Docs for PWAs  
  <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps>
* MDN Web app manifests   
  <https://developer.mozilla.org/en-US/docs/Web/Manifest>
* Web App manifest explanation by Google.  
  <https://web.dev/add-manifest/>