**How to Turn Your Website Into Progressive Web App (PWA) -** [Quokka Labs](https://medium.com/@quokkalabs135?source=post_page-----59702c353816--------------------------------) May 16, 2022

**Progressive web applications**(PWAs) turned into hot trends in the previous years, and it’s not surprising. PWA is easy to make, and they are dependable, quick, and engaging for the users.

The matter is that the [blend of a web application](https://quokkalabs.com/blog/a-comprehensive-guide-to-web-app-development/) and a mobile native application seems unique, and individuals didn’t dream of it before. Be that as it may, it has become conceivable, and a dynamic web application that is a mix of web and mobile applications turns out to be genuine.

Do you intend to make your own dynamic, progressive web application? Then, at that point, you ought to know the design of such an application first.

To ensure proper functionality, you should know how the advanced web functions and its key elements. Let’s discuss this in detail.

**Application shell in PWA**

With the help of an application shell, the progressive web application will work with superior execution. The application shell building process is simple, and a couple of lines will be expected to fabricate an application. The header and navigation bar are the principal components. The application shell can be accessible for the client and users without Internet association since it is stored only once.

Application shell should be available in advanced web applications. The navigation model’s remaining parts are unaltered, yet its content is refreshed and updated. An appropriate application shell shouldn’t over-burden the application. And it ought to load without lags.

**Service worker**

The [primary benefit of a progressive web application](https://quokkalabs.blogspot.com/2022/03/benefits-of-progressive-web-apps-pwas.html) lies in its ability to work with a decent Internet association and offline mode. Service worker innovation accomplishes it. The script checks the framework and figures out where it ought to download content — from the store or server.

A service worker assists an ever-evolving web application by working offline. So it is one of the fundamental parts such an application ought to incorporate. Remember that you want an HTTPS server to execute it.

**Manifest Record**

This record will assist you with making a splash screen so clients/users will want to see a fabulous image when your application is loading. A splash screen is vital for all applications, not advanced web applications.

*To convert the website into PWA, here are the things required –*

* The Manifest Addition
* Shifting to HTTP
* Adding a service worker
* Offline mode access
* Deploy & Testing

**1. The Manifest Addition**

A manifest is only a *JSON* document that describes your PWA. Things like your application’s name, language, and symbol go in. This data will tell programs how to show your application when it got saved as an alternate route.

We call “**manifest.json”** and connect from the <head> of your site.



An application manifest is a JSON document containing these data:

* The site’s official name & its shortened version of that name for icon.
* The theme and background shade of the website for OS combination.
* The beginning URL.
* Any symbols for your site to be utilized on the home screen.
* A human-readable description.

These data will be utilized as the OS-level metadata for your web application when it gets installed.

**2. Shifting to HTTP**

Progressive Web Apps should get served over a protected connection, so the HTTPS protocol is best. HTTPS scrambles the information clients ship off your server. And keeps interlopers from messing with their contact. Google additionally leans toward HTTPS and ranks them higher than non-secure contenders.

So then, to switch the site over to a progressive web application, you should add TLS (transport layer security) and make a shift to HTTPS. But How is this possible?

The TLS guarantees security while ensuring security. Alongside guaranteeing wellbeing against all unauthorized access.  
It is compulsory to benefit from an ‘SSL certificate’ to change to HTTPS. Ensure that you take a backup of all your data when you do this. Once done, update every one of the internal connections to HTTPS.  
If everything goes as expected, you’ll see a pleasant green lock symbol close to your URL, as shown below:

**3. Adding a service worker**

A Service Worker is a piece of Javascript that act as a mediator between program and host. It consequently introduces itself in upheld programs, can capture demands made to your site, and answer them in various ways.

You can set up another SW by making a Javascript document at the root index of your project. We can call this **sw.js**. The items in that record rely upon what you need to do — we’ll get to that in a moment.

We expect to involve this record as a Service Worker; We need to enlist it first. To let the browser know, we use this as a service worker. Like this:



The scope parameter characterizes which demands the SW ought to have the option to catch. It’s an overall way to the domain root. For instance, if you somehow happened to set this to/articles, you had some control over solicitations to yourdomain.com/articles/my-present yet not on yourdomain.com/contact.

**4. Offline mode access**

One of them is the capacity to reserve content, store it locally, and make it accessible when the user is offline. Regardless of whether they are on the website, this will tremendously affect page stacking time.  
You can characterize a rundown of assets to reserve when the service worker gets introduced — so a client/user doesn’t need to explore a page first for it to be stored. A good way that could look:

Text

Description automatically generated

You can essentially pass it a document, and it will consequently follow all changes and keep your Service Worker cache up to date. Since sw-precache coordinates into your site’s buildup. You can utilize trump cards to precache every one of the assets that match a particular pattern, as so:

You can have it precache each picture, *HTML, JavaScript, and CSS document* for the static sites. For websites that contain dynamic content or many enormous images not generally required. Precaching a “skeleton” subset of your site, frequently check out.

**5. Deploy & Testing**

Since you have moved to HTTPS, added service workers, and arranged the web application manifest to change your site to the progressive web application. There is no time like the present you begin applying a few checks. And tests to ensure that the transformation of your site to a web application is smooth and fruitful.  
  
Note that updated services get quickly and automatically initiated each time another page is loading.

Graphical user interface, application

Description automatically generated

The Chrome Lighthouse Extension is a famous testing tool used for PWA. Check PWAs for their performance, Accessibility, and consistency with the PWA spec.

It tests your site in various viewports and network speeds, time to paint, and other execution factors. And offers significant guidance for areas needing improvement. It’s an excellent benchmark for sites.

Now you know how to change over a Web application to PWA. And if you followed the step-by-step guide, you could turn your site into a Progressive Web App quickly!

**Conclusion:**

Time is changing continuously, and every technology & innovation has pushed one stage forward. Progressive web applications are the fate of programming improvement across the world. To change sites over completely to moderate web applications is an incredible move.

Thus, rather than attempting your hands to make an [application from a site](https://quokkalabs.com/web-application-development) and find an alternate outcome. Since you weren’t sure of the between-the-line components, give a job to a PWA development company specializing in the area. The expert will respond to every inquiry about switching a website into PWAs.

I hope the above steps will give you an exact idea of turning your website into