**PWAs**

**What are they?**

PWA stands for progressive web application.

The chief feature identifying a PWA, as opposed to a native desktop application or traditional web application, is that it attempts to deliver the look, feel, and functionality of a traditional mobile desktop application using frontend technologies (primarily HTML, CSS, and JavaScript)**(1)**. Fundamentally a PWA is a website; however, it must display and perform like a mobile application**(2)**

As in a traditional website, HTML is responsible for the content of the application; CSS, the design; and JavaScript, the behaviour; of these three, it is JavaScript which has the biggest part to play in delivering the native-app like functionality that is expected of these applications.

Another key expectation of a PWA is that it should be able to deliver at least some level of functionality offline; this is accomplished by caching data during periods of connectedness; the responsibility for this lies with a background program known as a service worker**(2).**

There are four pivotal features which an application requires to be considered a PWA**(2)**:

**A web app manifest:**   
This is a JSON (JavaScript Object Notation**(3)**)file providing data to the browser in use about the application and its expected behaviour; this should typically provide the name of the app and the URL at which to locate the app on its being launched**(4).**

**Service workers:**

As previously mentioned, these are event-driven**(5)** programs which run in the background of the application, separate from the script running in the user facing application (the application shell)**(8, 10)**; their main task is to coordinate network requests and cache data for periods of offline activity**(6)**.

**An icon:**

This allows the user to install the application on their home-screen; different native desktop environments will require icons in different formats; this must be considered during their creation**(2)**.

**Being served over HTTPS:**

Service workers only work on websites using the HTTPS protocol; this is as opposed to traditional websites which at least have the option of using HTTP**(7)**.

This, ideally, results in an application that is designed to run in all standards-compliant web browsers, possessing a uniform end-user experience across all of them.

**Advantages**

Given that most users can be assumed to have at least one standards-compliant web browser installed on their device, PWAs allow an application to be made as broadly available as possible. They are not coupled to the capabilities or conventions of any particular platform or hardware; do not require the user to sign up to any platform-specific app store**(13)**; and do not require the end user to engage in any lengthy installation, configuration, or updating process**(1, 2)**.

A well designed PWA has a realistic chance of working on any platform, desktop or mobile, with only negligible differences**(2)**. As websites, these applications retain the traditional advantages of websites: they are linkable, do not require the end user to store any great amount of data on their system, and their design can be tailored to the device in use using exactly the same techniques as would be used in traditional webpages.

The technologies used in creating PWAs are the same as those used in developing traditional web applications; therefore, it is simpler to find or train developers with the required skill set; they do not need to become familiar with an entirely new tech stack**(9)**.

As mentioned, the presence of service workers requires the use of HTTPS, thus making mandatory what is already best practice**(2)**; service workers cache resources, so that the application can continue running while offline**(1, 2)**.

Traditionally, native apps were generally able to outperform browser-based equivalents due to their having direct access to platform-specific features for boosting performance, being more easily integrated with other software native to the platform, and not having the web browser as an additional layer of abstraction between the application and the processor. This meant that browser-applications were generally permitted to handle only trivial functions and were usually a thin layer over much more involved server-side logic.  
  
Now, however, modern web browsers are able to run applications that would be too computation-heavy for many desktops only decades ago; if we consider the typical modern scenario of viewing 3d renderings of scenery and buildings in Google Maps whilst simultaneously streaming multimedia from several open tabs, we begin to appreciate the strength and breadth of modern browser performance. In light of this, PWAs seem like a promising and logical choice for anyone launching a new product.

**Disadvantages**

Despite what is written above, there are a number of issues that detract from PWAs; many of these stem from the same features as the above mentioned advantages.   
  
Running primarily within the browser, a PWA has limited access to platform-specific facilities and does not integrate as readily with software native to its platform; this is becoming less of an issue as browsers advance technologically; however, this still poses a problem with applications that need specific, particularly low-level, facilities**(11, 13)**.

Being built on top of a web browser, PWAs are inherently more high-level than many equivalent desktop applications; this is not as much of an issue as it used to be; however, there are still applications where the speed and precision of low-level code are required; these are poor candidates for PWA development, and generally require a more traditional approach**(11)**; it is likely that computation-intensive tasks of the kind found in the automobile industry, for instance, will remain strictly the domain of lower-level code for the foreseeable future. Particularly, when we consider the experience of mobile users, PWAs can have a considerably higher impact on battery consumption**(11)**, which could be off-putting to potential customers.

While browser support is not as much of a concern as it used to be, it is still difficult to be certain that a PWA will enjoy complete support across all devices; indeed, some vendors are slightly averse to PWAs as a concept due to the potential impact to their business model**(12)**; although, there are indications this is changing**(12 - see 2017 update, 13)**.

Whilst there is some convenience attached to bypassing the app store installation route, PWAs can be offered anywhere on the internet, with the end user having to decide whether or not the source is trustworthy; this has security implications far less likely to arise within officially sanctioned app stores**(13)**.

**Real-World Examples**(17)

* Alibaba, the world’s largest online B2B trading platform, designed a new PWA and subsequently enjoyed a 76% increase in online conversions**(14)** - they used the high-profile JS framework, vue.js**(15)**.
* Debenhams constructed a PWA with the platform, Mobify**(16)** - since doing so, they have seen a 40% increase in mobile revenue and a 20% increase in conversions**(16)**.
* Uber, one of tech’s most widely recognized names, constructed a PWA, m.uber, to offer an app-like experience to end-users owning devices not supportive of their native client; this demonstrates the increase in usership that can be gained by conversion to PWA; this was built using the framework, Fusion.js**(18)**.
* Forbes launched a PWA in 2017; the increased speed and user-friendliness of this application resulted in their readers’ article completion rate growing to 6 times its previous level and their average session time increasing by 100%**(19)**; their user engagement rate was doubled**(17)**.

**References**

1. [**https://en.wikipedia.org/wiki/Progressive\_web\_application**](https://en.wikipedia.org/wiki/Progressive_web_application)
2. [**https://hackernoon.com/everything-you-need-to-know-about-progressive-web-app-pwa-6524edbb0c57**](https://hackernoon.com/everything-you-need-to-know-about-progressive-web-app-pwa-6524edbb0c57)
3. [**https://en.wikipedia.org/wiki/JSON**](https://en.wikipedia.org/wiki/JSON)
4. [**https://web.dev/add-manifest/**](https://web.dev/add-manifest/)
5. [**https://en.wikipedia.org/wiki/Event-driven\_programming**](https://en.wikipedia.org/wiki/Event-driven_programming)
6. [**https://love2dev.com/blog/what-is-a-service-worker/**](https://love2dev.com/blog/what-is-a-service-worker/)
7. [**https://www.smashingmagazine.com/2018/11/guide-pwa-progressive-web-applications/**](https://www.smashingmagazine.com/2018/11/guide-pwa-progressive-web-applications/)
8. [**https://developers.google.com/web/fundamentals/primers/service-workers**](https://developers.google.com/web/fundamentals/primers/service-workers)
9. [**https://hackernoon.com/the-advantages-of-progressive-web-apps-pwa-versus-native-apps-5o2aq308m**](https://hackernoon.com/the-advantages-of-progressive-web-apps-pwa-versus-native-apps-5o2aq308m)
10. [**https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps**](https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps)
11. [**https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps**](https://clutch.co/app-developers/resources/pros-cons-progressive-web-apps)
12. [**https://love2dev.com/blog/apple-encouraging-progressive-web-apps-by-rejecting-apps/**](https://love2dev.com/blog/apple-encouraging-progressive-web-apps-by-rejecting-apps/)
13. [**https://avengering.com/en/why-use-the-pwa-what-are-advantages-and-disadvantagesof-pwa-part-2/**](https://avengering.com/en/why-use-the-pwa-what-are-advantages-and-disadvantagesof-pwa-part-2/)
14. [**https://developers.google.com/web/showcase/2016/alibaba**](https://developers.google.com/web/showcase/2016/alibaba)
15. [**https://madewithvuejs.com/alibaba**](https://madewithvuejs.com/alibaba)
16. [**https://www.thinkwithgoogle.com/intl/en-gb/success-stories/global-success-stories/debenhams-progressive-web-app-boosts-speed-conversions-and-revenue/**](https://www.thinkwithgoogle.com/intl/en-gb/success-stories/global-success-stories/debenhams-progressive-web-app-boosts-speed-conversions-and-revenue/)
17. [**https://medium.com/progressivewebapps/best-pwa-examples-for-your-inspiration-261bcb3fab47**](https://medium.com/progressivewebapps/best-pwa-examples-for-your-inspiration-261bcb3fab47)
18. [**https://eng.uber.com/web-booking-flow/**](https://eng.uber.com/web-booking-flow/)
19. [**https://www.e-point.com/\_fileserver/item/1501040**](https://www.e-point.com/_fileserver/item/1501040)