uve Follow
Developer advocate, Samsung Internet. Programmer, Game Developer, Event Organiser. I make g...
Jul 25 · 7 min read

# A beginner's guide to Service Workers

This guide is aimed at people who have some experience with HTML, CSS, and JavaScript. It covers the basics of service workers and how to use them without going into too much detail, with the aim of letting you build a prototype as quickly as possible.

Once you have a basic demo and understanding then you can look at other resources and the theory of how things work as you need them.

## What is a service worker?

A service worker is a JavaScript file you add to your website. You can use it to do offline and background stuff, such as make your website display pages when offline or display push notifications when your site is not open.

## What can they do?

Some examples of what you can do by adding a service worker to your site include;

- **Display an offline page** when the user does not have internet access.
- **Store pages offline** as the user visits them, so they can view them when they are offline.
- **Store content in the cache** as soon as the site is loaded, so all or selected pages can be viewed offline, even if the user had not previously visited them.
- If used with a web app manifest it will be seen by mobile web browsers as a **Progressive Web App,** this means users will be prompted to add it to their phones homescreen.
- Using **push notifications**, your website can send the user notifications even if it isn't open. You may have seen this used by Mastodon, Twitter or Facebook which send you a push notification when you get a new notification or message on their site.

- Send messages and upload/download content even if the site isn't open. **Background Sync** lets it do tasks only once the user has a stable connection.

# Adding a service worker to your project

Implementing a service worker involves adding some code to your HTML or existing JavaScript and adding in a new JavaScript file to your project that will act as the service worker itself. The code in the HTML will tell the browser that you have a service worker and will "*register*" it.

I'll introduce you to two different ways you can get service worker code for your project.

If you are following along and using this guide as a tutorial; now would be a good time to make a website that you want to add a service worker too. If you don't have a website built yet, don't fret, you can just download a template from online. I personally like the ones from HTML5 UP and Start Bootstrap.

The website I am using for this demo is SWQuiz. It is based of a codepen project.

## PWA builder

PWA builder is a nice online tool that helps create Progressive Web Apps, for now, we just want to use the service worker generator parts of it.

Open it up, it will take you to a website manifesto generator, you can build one if you'd like or just skip to the service worker section. Here you will have a few options to pick from.

For each option, you will need to add the 'Code for website' to your HTML head in between some `<script> </script>` tags and add the service worker to the project as a JavaScript file, make sure it is named exactly the same thing as in the HTML. In most cases, you will also need to edit the HTML or Service Worker code slightly or add in other files. I'll go through that when describing each option.

Note: the service worker can only access folders below it, so it is probably best to keep it in the root of the project (the top most folder).

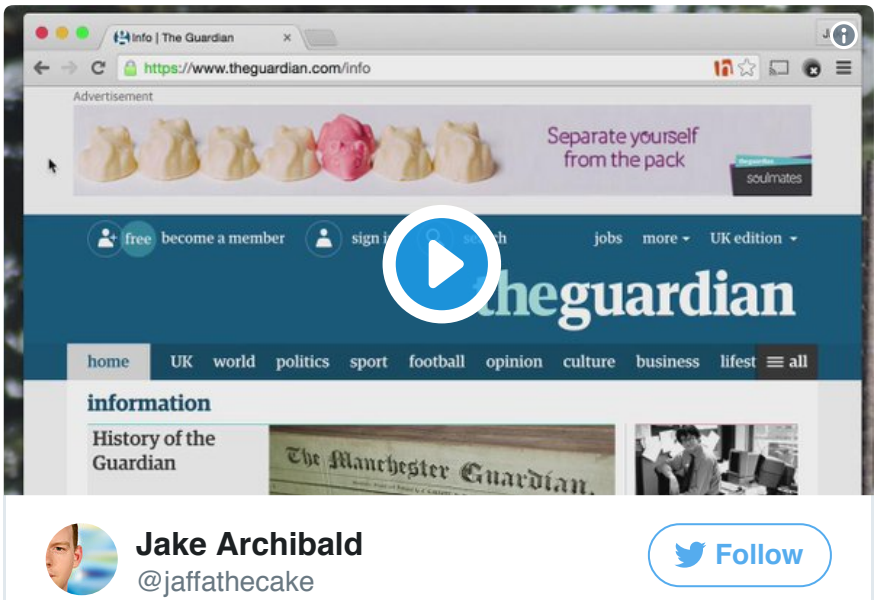
**Offline Page**
This option loads in a specific "you are offline" page when the user does

not have internet access.

Once you add the code to your project you will also need to create a file called `offline.html` which will be the page that comes up when the user does not have internet access.

If you load your website, then go offline, when you navigate to a different page you will now see your `offline.html` page rather than your browser telling you that you are not connected to the internet. This is useful as you can still give some content to your users when they are offline.

I made a simple demo using this service worker, if you load up the page, go offline then refresh, you can still see the main page. If you click on the link on the top of the page it will take you to a page saying you are online or offline. You can check out all the code for it on GitHub.




The Guardian added a crossword to their site for readers who lose internet access during their visit.

**Offline copy of pages**
This option saves pages to the cache as users visit them. So if the user goes offline they can still go back and view pages they have previously been on.

Like the other options, you just put in the service worker and website code into your project and you're good to go. There is no extra editing or new pages to add.

This is my favourite option as it doesn't require any extra set up, but you should do some testing to make sure it works properly. I've used this

when turning a single page site into a Progressive Web App. But because it does not have a timeout it could take ages to decide to use the cache when on a slow internet connection.

**Offline copy with Backup offline page**
This option combines both the previous ones into one service worker. If the user has visited your site then returns when offline they will be able to visit pages they have previously seen. If they get to a new page, your custom offline page will be displayed instead.

Add the code to your project and make sure to create and add in a page called `offline.html` which will be shown to the user when they don't have internet access and are visiting pages that have not been cached.

**Cache-first network**
This service worker will download all the pages and files you tell it to and save them for later. When a user visits one of those pages it will load from the cache. This normally means the page will load fast and work offline. If online, the service worker will also check if it has the latest version of the page on load.

After adding the code to your project you will need to add in the files you want to cache to an array in the service worker. Mine looked something like this.

```
var precacheFiles = [
 "index.html", "css/style.css", "js/index.js"
 ];
```

This benefits your user as after the initial load, cached content will load very fast, content will be kept updated if they have an internet connection and if offline they can still view pages.

I added this service worker to this page, you can see the code for it on GitHub.

## Service Worker Toolbox

Service Worker Toolbox is a collection of tools that help make building service workers easier. Learning how to use it should be faster than building your own from scratch.

To get you started with it I'll give you some code to copy taken from one of my previous projects. First, copy this into the head of your HTML file. This will register the service worker `sw.js` .

```
<script>
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/sw.js').then(
      function(registration) {
        // Registration was successful
        console.log('ServiceWorker registration successful
with scope: ', registration.scope); },
      function(err) {
        // registration failed :(
        console.log('ServiceWorker registration failed: ',
err);
      });
  });
}
</script>
```

Then you need to add this file to your project. You don't need the whole of the service worker toolbox in your project, just that main JavaScript file.

Then you need to create a service worker and put this code in it. I named mine sw.js but feel free to call yours whatever you want. (If you do use a different name update the code that goes in the HTML, or the service worker won't be registered)

```
'use strict';
importScripts('sw-toolbox.js');
toolbox.precache(["index.html","style/style.css"]);
toolbox.router.get('/images/*', toolbox.cacheFirst);
toolbox.router.get('/*', toolbox.networkFirst, {
networkTimeoutSeconds: 5});
```

You have to edit `toolbox.precache(` to include all of your website files that you want to be cached straight away.

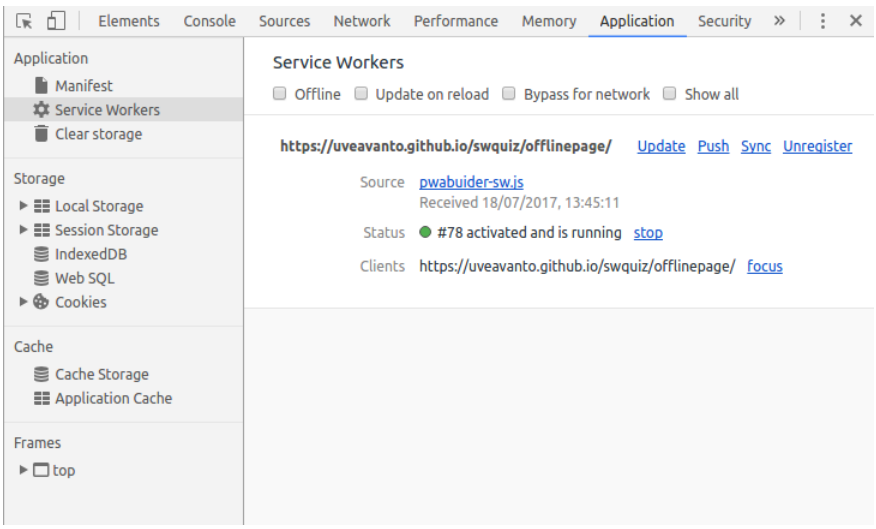
Now you have a service worker that will precache content, and if it get a response back for 5 seconds will show the content in the cache.

You can do a lot more with service worker toolbox than just that, but as you can see you can do a lot with relatively little code compared to the service workers from PWA Builder.

## Testing and Debugging your service worker

If you are using Chromium or Google Chrome, you already have the developer tools you need. Firefox also has its own tools, but for this guide, I'm going to focus on the ones in Chromium as they are the ones I am most familiar with.

When on a page you want to test, right click and select 'Inspect'. The developer tools will then open normally at the bottom or the right of the window and show you the 'Elements' tab. There is a lot of stuff in the dev tools, for now, you just want to play with service worker related stuff, so select 'Application' from the bar at the top, it may be hidden under `>>` . w in




Here is where you can play about with your service worker. The main features I use here are; the **Offline** tick box to make the site think you are offline, the **Unregister** link next to the service worker that deletes it and the **Clear storage** option whenever I make any big changes to make sure it updates everything. Checking through **Storage** is also useful to make sure the service worker is caching pages.

When testing websites with service workers you need to run them on a server, rather than just opening up the html file in a browser. You can do this locally using html-server or web server for chrome or you could host the site online using GitHub Pages.

## Further reading/more info

If you want to build a Progressive Web App, I wrote a guide to making one from scratch.


If you want to learn more about service workers, this talk by Phil Nash goes into a little more depth than this guide and would make for a good next step.