How to make a Progressive Web Application

The goal is to embed a web app access inside a headless browser. Following method comes from `https://alicia-paz.medium.com/make-your-rails-app-work-offline-part-1-pwa-setup-3abff8666194` .

Needed ingredients

- A service worker : has to be avalaible at `/service-worker.js` . It is a small application that runs in parallele to your Rails app. It is a sort of proxy between app and headless browser to provide functionnalities like notifications.
- A app manifest : has to be avalaible at `manifest.json` . It tells the browser how your PWA should display within the user device. It responsible to make your app *installable* and thus to make it *look and feel* like a native app.

To make routes to serve these files, a solution is to provide a controller because it is the rails way of serving files.

Create a service worker controller

```
# app/controllers/service_worker_controller.rb
class ServiceWorkerController < ApplicationController
  protect_from_forgery except: :service_worker # has to be served anyway.
  skip_before_action :authenticate_user! # if devise ! Replace 'user' by devise model name

  def service_worker
  end

  def manifest
  end
end
```

Routes

```
# config/routes.rb
  get "/service-worker.js" => "service_worker#service_worker"
  get "/manifest.json" => "service_worker#manifest"
```

Add your service worker and manifest

As we create routes and controller actions, we serve these files from a view !

manifest.json

```
// app/views/service_worker/manifest.json.erb
{
  "short_name": "YourAppName",
  "name": "YourAppName",
  "id": "/",
  "icons": [
    {
      "src": "<%= image_path 'your-fancy-icon.png' %>",
      "sizes": "(144x144 256x256 512x512",
      "type": "image/png",
    }
  ],
  "start_url": "/",
  "background_color": "#000000",
  "display": "standalone",
  "scope": "/",
  "theme_color": "#000000"
}
```

Then we require the manifest in the application layout adding support for CORS with the crossorigin attribute.

```
<!-- app/views/layout/application.html.erb ABOVE stylesheet link -->
<link rel="manifest" crossorigin="use-credentials" href="/manifest.json" />
```

service-worker.js

```
// app/views/service_worker/service_worker.js (minimal content)
function onInstall( event ) {
  console.log( '[ServiceWorker]', "Installing!", event );
}
function onActivate( event ) {
  console.log( '[ServiceWorker]', "Activating!", event );
}
function onFetch( event ) {
  console.log( '[ServiceWorker]', "Fetching!", event );
}

self.addEventListener( 'install', onInstall );
self.addEventListener( 'activate', onActivate );
self.addEventListener( 'fetch', onFetch );
```

Write the companion JS and put it in the asset pipeline

This file tells your application when and from where to load your service worker.

```
// app/javascript/custom/companion.js
if ( navigator.serviceWorker ) {
  navigator.serviceWorker.register( "/service-worker.js", { scope : "/" } )
    .then( () => navigator.serviceWorker.ready )
    .then( (registration) => {
        if ( "SyncManager" in window ) {
            registration.sync.register( "sync-forms" );
        } else {
            window.alert( "This browser does not support background sync." )
        }
    })
    .then(
      () => console.log( "[Companion]", "Service worker registered!" )
    );
}
```

For rails to see this file, we import it in `application.js` :

```
// app/javascript/application.js
import "custom/companion"
```

And, within rails7 defaut asset pipeline, we also need to pin the module in the import map :

```
# config/importmap.rb
pin_all_from "app/javascript/custom", under: "custom"
```

Enjoy !

Start the server and go to root url. On a mobile device, your should be prompted to install the app. After accepting, your will see your fancy icon added on your home screen. Clicking it will launch a standalone browser window.