

# ASSIGNMENT 3 : PARTS 4 + 5 (OF 5)

Progressive Web Apps  
2023



You must create a Progressive Web App with the functionality described in this document.

The PWA will be hosted on a server provided for you and will be served via HTTPS , it must work offline, and be installable.



## Features of Assessment:

Work Offline with Service Workers

Cache Management

Multithreaded with Web Workers

DOM Scripting

Web Services

Objects / Closures



**Part 1:** Create an App Shell (can just update Assignment 1)

**Part 2:** Create a page to Search Flickr

**Part3:** Create a page to Search a large JSON object

**Part 4:** Make your site Installable as app (i.e. a PWA)

**Part 5:** Implement the provided Caching Policies



## **Part 4: Make your PWA Installable as app**



Arrange for your PWA to be installable on devices.

I.e. It should have:

- a responsive icon

- an official app name

- a splash/loading screen

It should display without browser chrome.



**Note:** PWA manifests don't set icons in some versions of iOS

You can use the older **apple-touch-icon** link instead (you will need several of these).

```
<link rel="apple-touch-icon" sizes="512x512" href="polaroid-512.png">
```



You can find sample icons (for the purposes of this assignment) on sites like:

[iconfinder.com](https://iconfinder.com)

Each icon is usually available in different sizes.



## **Part 5: Caching Policies**



You will need to write a service worker (shared by the two files) that will manage your caching policy.

The next pages contain a description of the caching policy that you must implement.



The **Caching Policy** is essentially a description of what the Service Worker will do in response to a request from one of your pages.

It describes where it will look for the response (the cache, the network, or generate the response itself), as well as the priority it gives to each source (e.g. search the cache first, if its not there check the network).

It also specifies when items are added to the cache (and whether some items are to be cached at all).



Different types of resources will require a different policy.

i.e.

**App Shell Files** (HTML, JavaScript, CSS and images, etc)

**Flickr Search Result** (i.e. JSON-P from flickr.com)

**movieobj.js** (JSON-P file containing script data)

**Images** from Flickr



## The Policy



## **App Shell Files** (HTML, JavaScript, CSS and images, etc)

Basic files needed for your app shell to function should be immediately cached when the service worker installs.

HTML, CSS, images, .js, fonts, etc.

**Cache Policy:** Cache on installation / for every consequent request you should check the cache first, then the network if it wasn't in the cache (if found online then add it to the cache).



**Flickr Search Results** (i.e. JSON-P from [flickr.com](https://www.flickr.com))

**Don't cache** the Flickr Search results (i.e. the **JSON-P** response, we will be caching the actual images)

**Cache Policy:** Check the Network first, if it can't be accessed then just send back fallback content indicating the app is offline (i.e. some JSON describing what happened).



**movieobj.js** (JSON-P file containing script data)

**Don't cache the movieObj.js JSON-P file** (i.e. film script data)

**Cache Policy:** Check Network first, if it can't be accessed then send back fallback content indicating app is offline.



**Images** from Flickr

**Cache** the Flickr images

**Cache Policy:** Check in the cache first, if they are not in the cache then check the Network (and if found online add it to the cache)



Occasionally if you can't find a file in the cache or online you want to send back data your script will understand.

In this assignment we have cases where a JSON/JSON-P resource is being requested. If not found you can easily generate a JSON response in the ServiceWorker\* and send it back to the requesting page. This JSON should contain data that the page can use to detect something went wrong (e.g. it was offline).

(\*ServiceWorkers also let you send back HTML pages, alternative images, etc.)



E.g. here we return our own JSON-P file instead of the one we failed to read online.

```
return responseFromFetch.then (

    function() {      <... return the resource ... >      }

).catch (

    function() {      return new Response(

        "showImages({offline: true})",

        {headers: {"Content-Type": "text/javascript"}}

        );

    }

);
```



The response you send back should contain the JSON and set its **content-type** to **text/javascript**.

JSON-P response



```
return new Response(
```

```
    "showImages({offline: true})",
```

Set the header



```
{headers: {"Content-Type": "text/javascript"}}
);
```



# Testing & Development



You can use tools to help develop your PWA.

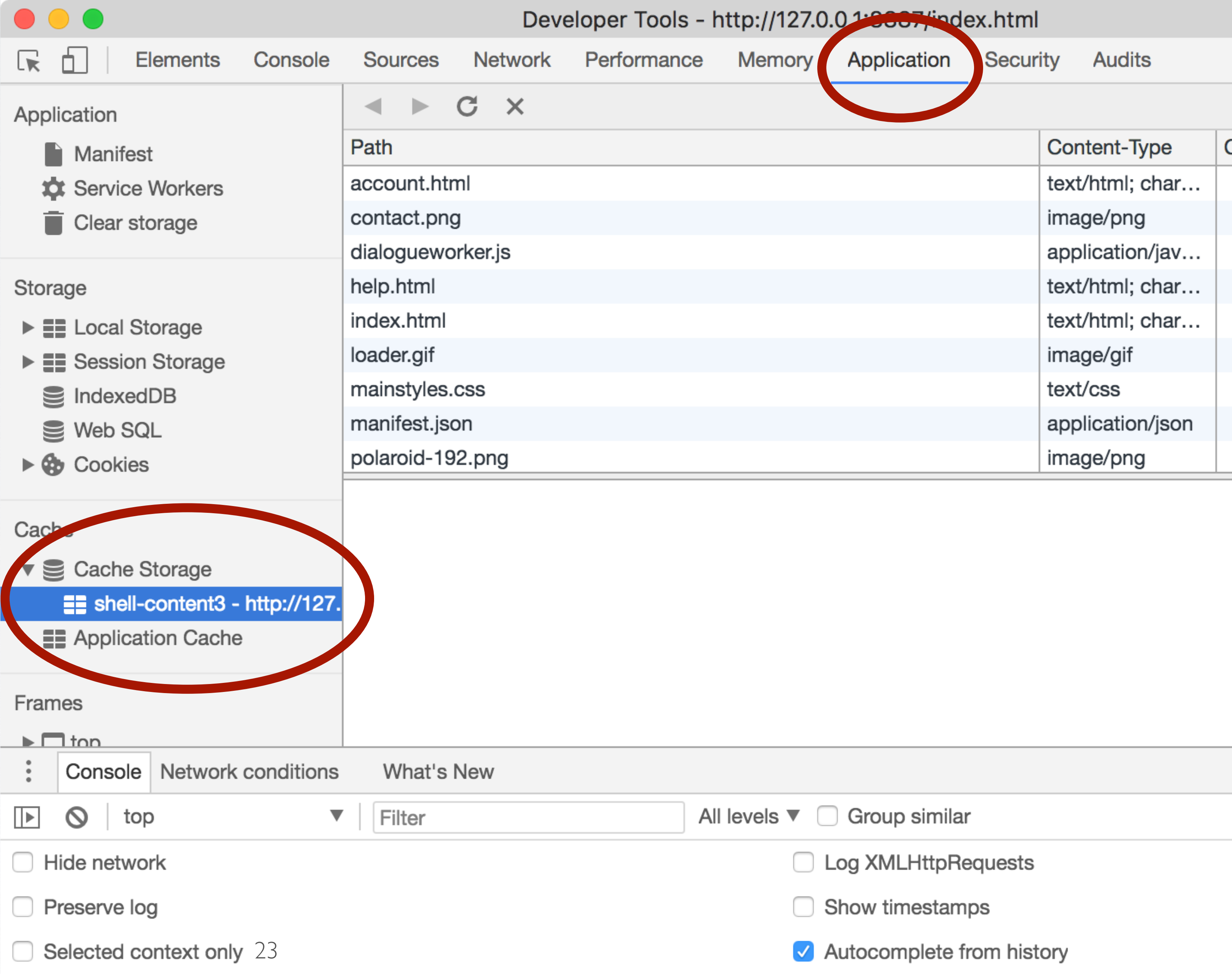
For example, Chrome has features that allow you simulate various network conditions.

You will find them among the regular developer tools.

(Other browsers have similar tools.)



Here you can examine your cache (and delete the entire cache or individual files within it which is useful when you want to test the initialisation phase multiple times).



The screenshot shows the Chrome Developer Tools interface with the 'Application' tab selected. The left sidebar is divided into sections: 'Application' (Manifest, Service Workers, Clear storage), 'Storage' (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), 'Cache' (Cache Storage, **shell-content3 - http://127.0.0.1:8087/**, Application Cache), and 'Frames' (top). The 'Cache' section is circled in red. The main panel displays a table of cached resources for the selected cache.

Path	Content-Type
account.html	text/html; char...
contact.png	image/png
dialogueworker.js	application/jav...
help.html	text/html; char...
index.html	text/html; char...
loader.gif	image/gif
mainstyles.css	text/css
manifest.json	application/json
polaroid-192.png	image/png

At the bottom, there are checkboxes for 'Hide network', 'Preserve log', 'Selected context only', 'Log XMLHttpRequests', 'Show timestamps', and 'Autocomplete from history' (checked).



You can emulate being offline.

The screenshot shows the Chrome Developer Tools interface with the Network tab selected. Two red circles highlight the 'Network' tab and the 'Offline' toggle switch. The 'Offline' toggle is currently turned off, and the 'Online' toggle is turned on. The 'Disable cache' checkbox is checked. The 'Filter' field is empty. The 'Hide data URLs' checkbox is unchecked. The 'All' filter is selected. The 'XHR' filter is selected. The 'JS' filter is selected. The 'CSS' filter is selected. The 'Img' filter is selected. The 'Media' filter is selected. The 'Font' filter is selected. The 'Doc' filter is selected. The 'WS' filter is selected. The 'Manifest' filter is selected. The 'Other' filter is selected. The 'Waterfall' view is selected. The '200.00 ms' total time is shown. The '▲ 3' icon is visible.

Name	Status	Type	Initiator	Size	Time	Waterfall	200.00 ms	▲ 3
search2.html	200 OK	document	Other	(from Serv...	124 ms 119 ms			
mainstyles.css	200 OK	stylesheet	search2.html Parser	(from Serv...	6 ms 5 ms			
css?family=Fjalla+One Oswald fonts.googleapis.com	200	stylesheet	search2.html Parser	(from Serv...	7 ms 6 ms			
mainstyles.css 127.0.0.1	200 OK	fetch	service1.js:98 Script	6.0 KB 5.8 KB	25 ms 19 ms			
css?family=Fjalla+One Oswald			service1.js:98	5.8 KB	25 ms			



While developing your app you should disable the cache (this is the regular browser cache and not the cache used by your Service worker)

You can emulate slow network speeds to better see how your app operates. You can set this to *Slow 3G* or equivalent.

A screenshot of the Chrome DevTools Network panel. The 'Network' tab is selected and circled in red. A red arrow points from the text 'While developing your app you should disable the cache' to the 'Disable cache' checkbox, which is checked. Another red arrow points from the text 'You can emulate slow network speeds...' to the 'Slow 3G' option in the settings menu. A third red dashed arrow points from the 'Slow 3G' option in the settings menu to the 'Slow 3G' option in the Network panel toolbar. The toolbar also shows 'Offline' and 'Online' options. Below the toolbar, there is a filter bar with 'All' selected, and a list of network requests. The first request is 'search2.html' with a status of '200 OK' and a type of 'document'. The second request is 'mainstyles.css' with a status of '200 OK' and a type of 'stylesheet'. The third request is 'css?family=Fjalla+One|Oswald' with a status of '200' and a type of 'stylesheet'. The table also shows the initiator, size, time, and waterfall for each request.

Name	Status	Type	Initiator	Size	Time	Waterfall	200.00 ms	▲ 3
search2.html	200 OK	document	Other	(from Serv...)	124 ms 119 ms			
mainstyles.css	200 OK	stylesheet	search2.html Parser	(from Serv...)	6 ms 5 ms			
css?family=Fjalla+One Oswald fonts.googleapis.com	200	stylesheet	search2.html Parser	(from Serv...)	7 ms 6 ms			



Usually you have to refresh a page two or more times for a change in a service worker to take effect (by design). Chrome provides a way to get immediate updates when testing your code.

The screenshot shows the Chrome DevTools interface with the 'Application' tab selected. In the left sidebar, the 'Service Workers' option is highlighted. The main panel displays the 'Service Workers' section for the local host '127.0.0.1 - deleted'. The 'Update on reload' checkbox is checked. Below this, a service worker entry is shown with the source 'service1.js' and a status of '#1287 is redundant'. At the bottom, there are input fields for 'Push' (containing 'Test push message from DevTools.') and 'Sync' (containing 'test-tag-from-devtools'), each with a corresponding button.

Developer Tools - http://127.0.0.1:8887/search2.html

Elements Console Sources Network Performance Memory **Application** Security Audits

Application

- Manifest
- Service Workers**
- Clear storage

Storage

- Local Storage
  - http://127.0.0.1:8887
- Session Storage
- IndexedDB
- Web SQL
- Cookies

Cache

### Service Workers

☐ Offline ☒ Update on reload ☐ Bypass for network

**127.0.0.1 - deleted** [Update](#) [Unregister](#)

Source [service1.js](#) ✖ 24

Received 23/04/2018, 23:45:03

Status ● #1287 is redundant

Push

Sync



Some features of PWAs require HTTPS. (Frequently this requirement is waived when you use **localhost** as your server for development purposes).

However, you will be provided secure hosting for your PWA.  
(I will confirm your logins – once I get them ready from the IT dept.)

Make sure you use https when accessing your pages.

i.e.

`https://webdevcit.com/.....`

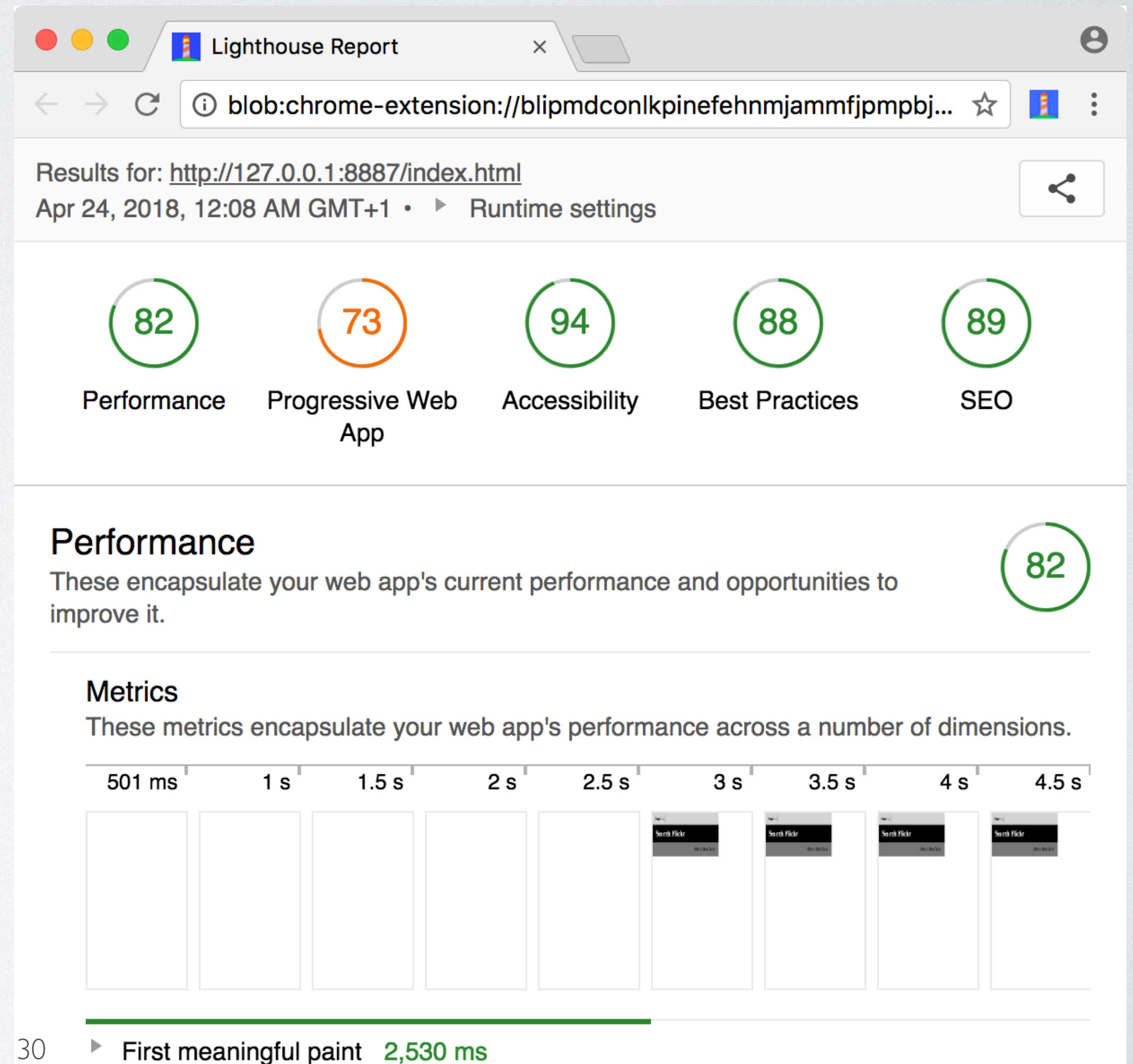
not

`http://webdevcit.com/.....`



<https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmmfjpmphbjk?hl=en>

Lighthouse can check the performance of your PWA for you (although not a requirement for this assignment).





# Submission



Your app should be available online on the server provided for you.

Your final PWA should be installable from that server.

You must also submit the code in a zipped folder to **Canvas**.



Your main page should be called **index.html**

Make sure both pages are reachable from this page.

You should also email me a URL to your PWA on the server. Your email should have the subject:

PWA 2023: Assignment 3 Submission

See also the **Assignment Guideline** document.

All code should be your own. You cannot use frameworks.