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

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)

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

Set the header

```
return  new Response(

        "showImages({offline: true})",

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

Testing & Development

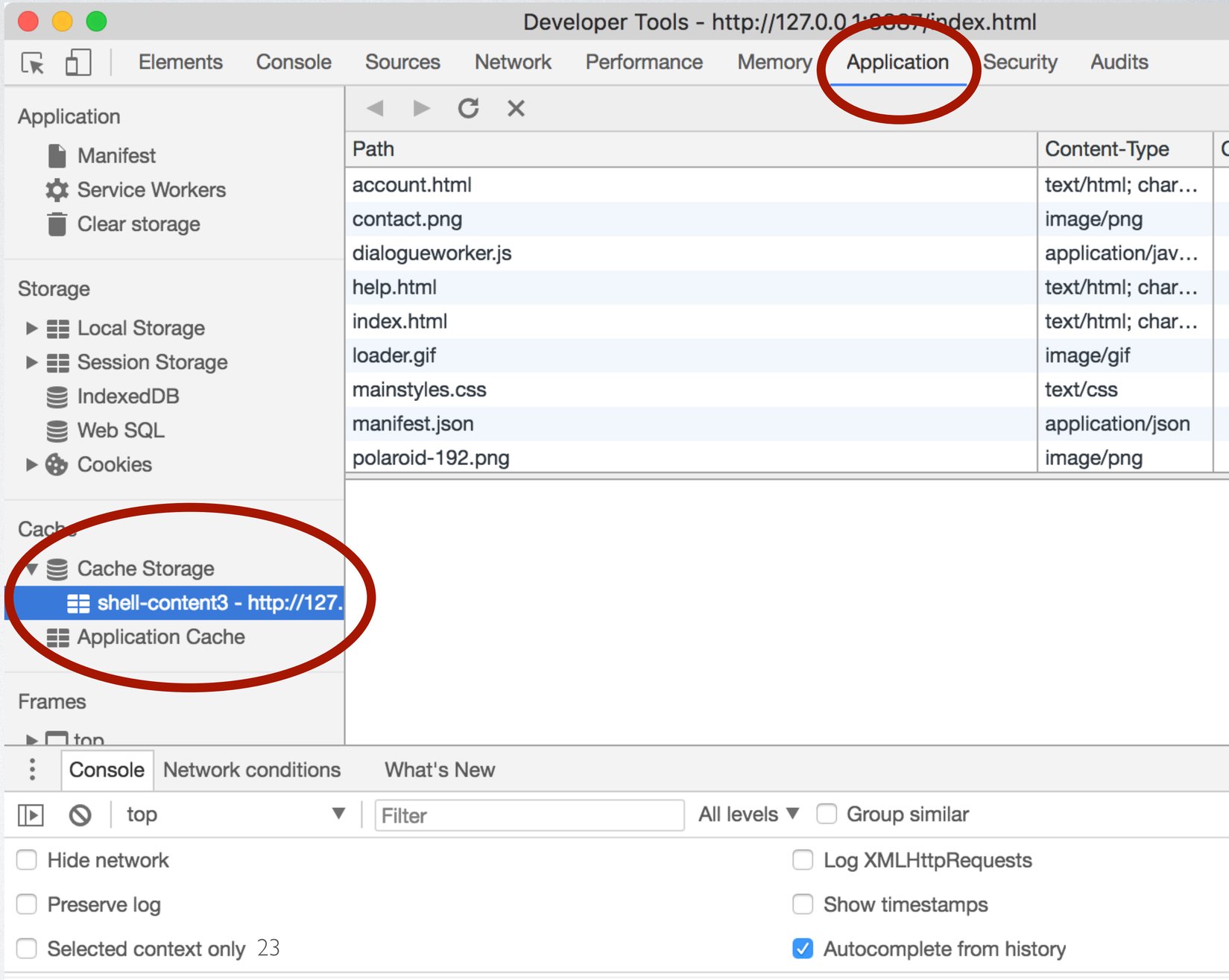
You can use tools to help develop your PWA.

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

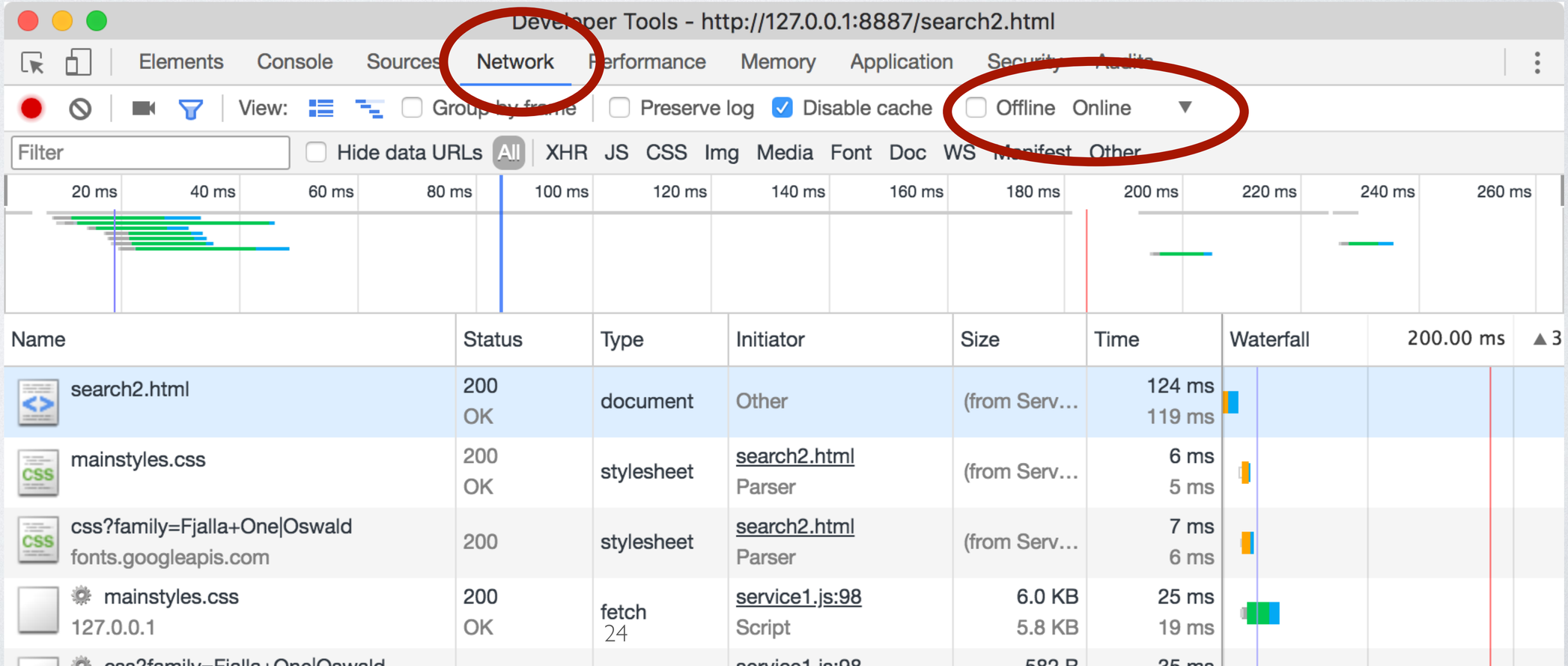
You will find them among the regular developer tools.

(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).

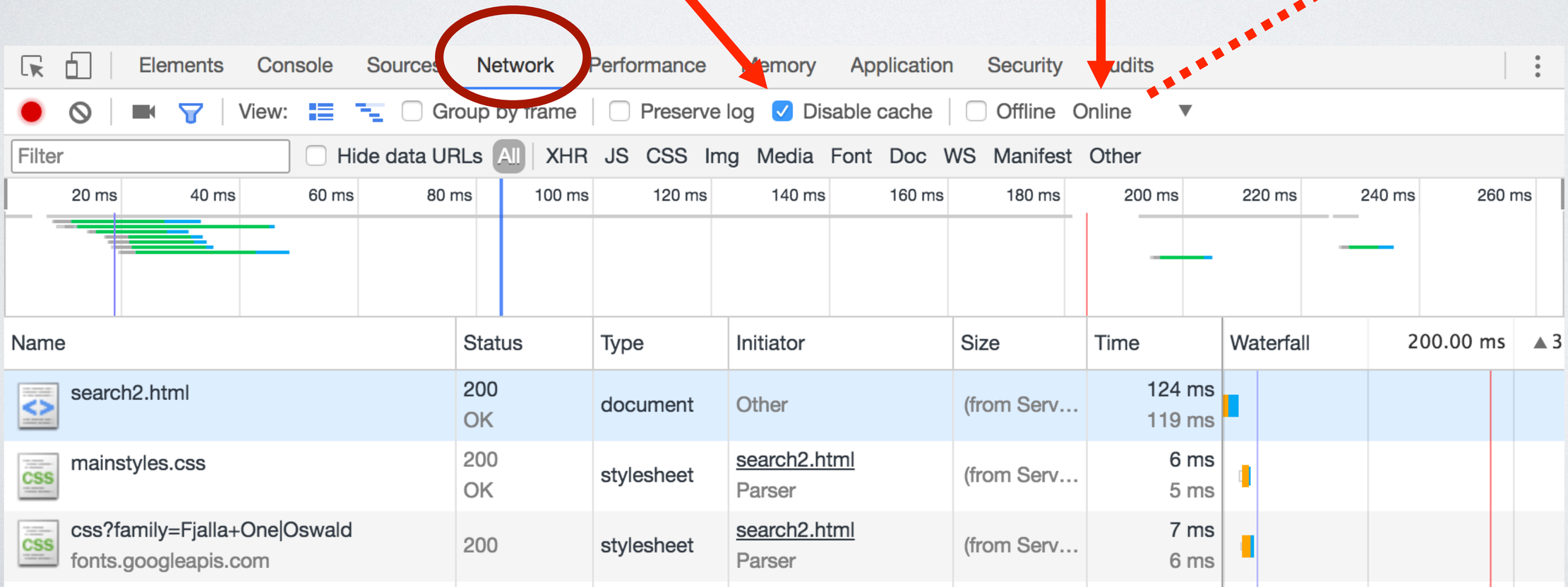


You can emulate being offline.

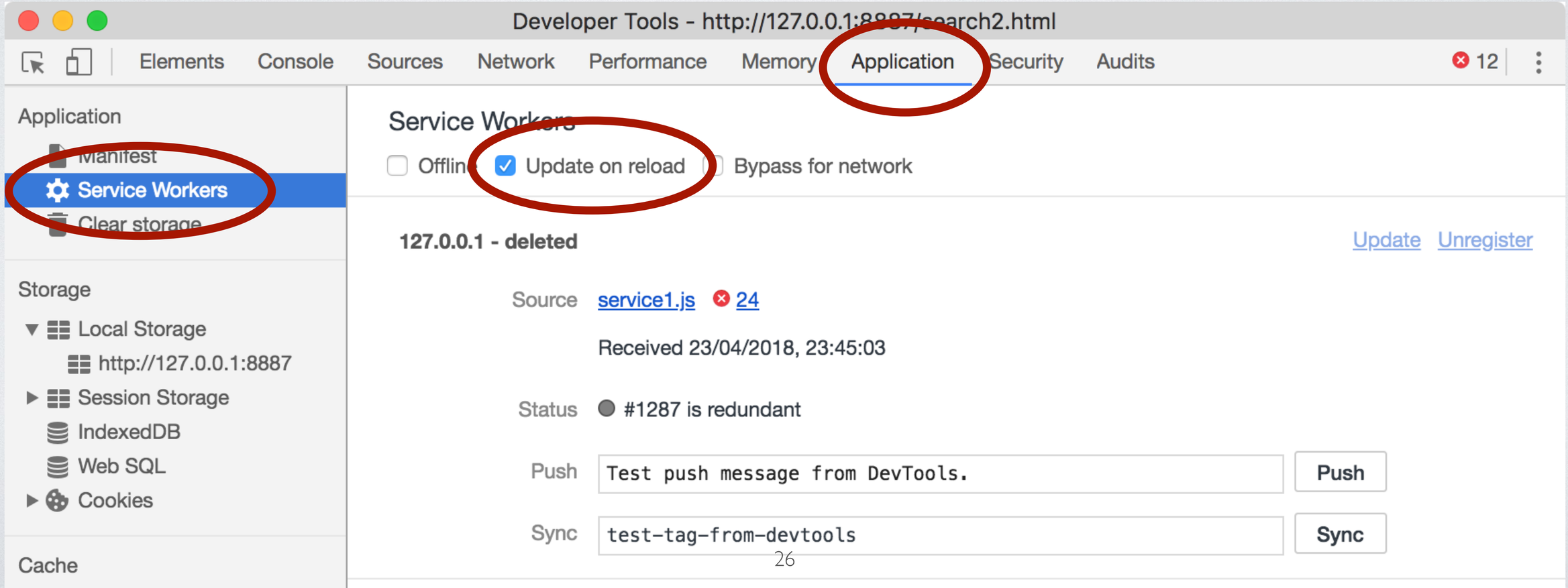


While developing your app you should disable the cache (this is the regular browser cache and not the cache used by your Service worker)

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



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.



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.

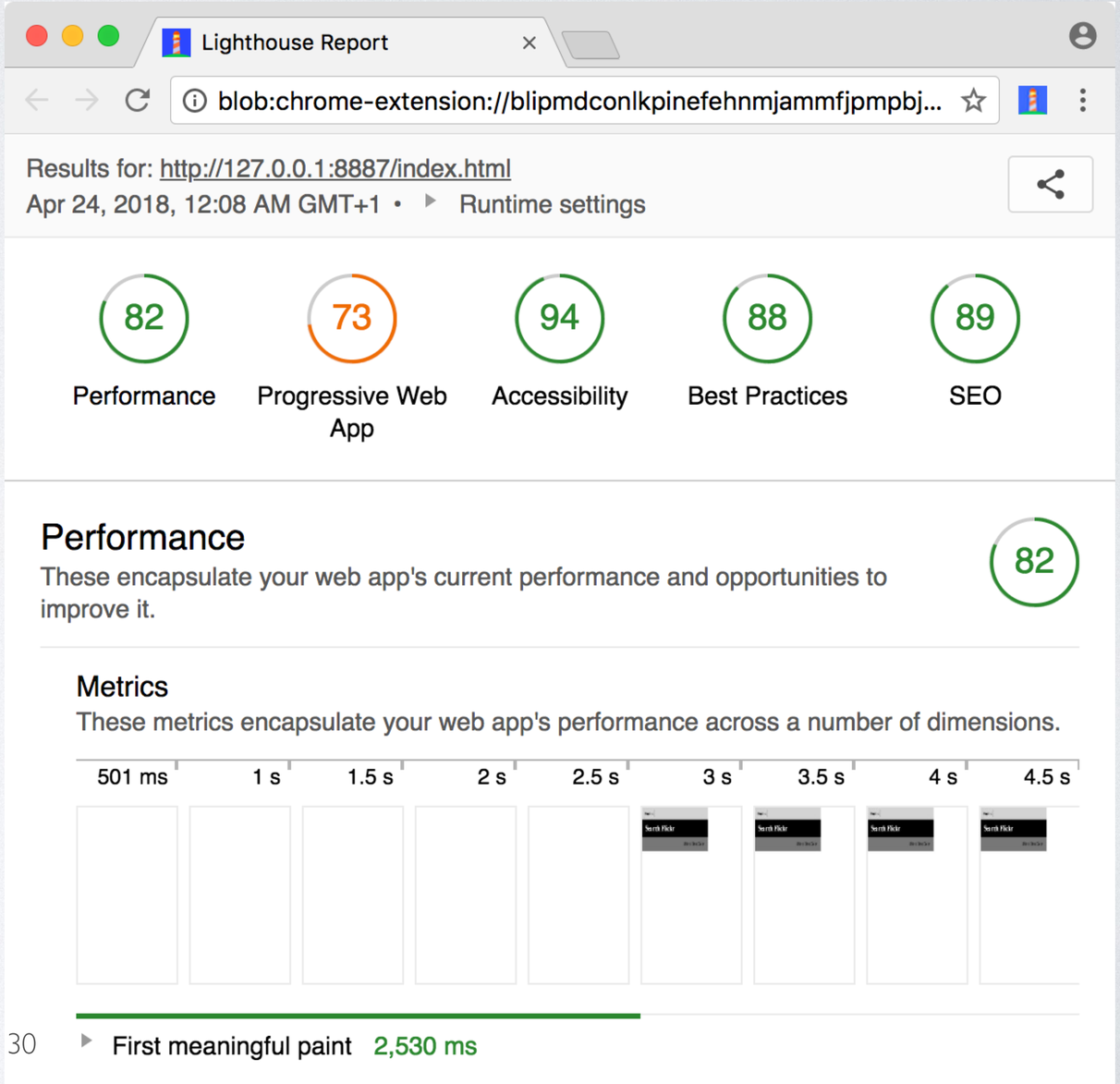
`https://webdevcit.com/.....`

not

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

https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=en

Lighthouse can check the performance of you 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.