BUSINESS ACADEMY AARHUS
UNIVERSITY OF APPLIED SCIENCES

Progressive Web App

Add to home screen

Progressive Web Apps (PWAs) are web apps that use service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience on par with native apps.

Image of Thomas Ulrich from Pixabay

Add to home screen (A2HS) - WD

The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load it's assets through a Service Worker.

Usage: % of all users ? — Global: 74.3% + 14.87% = 89.18%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE
	12-16		2-75		
4-38	17-18 [1]		76-85 [3]		
39-104	79-104	3.1-15.6	86-104	10-90	6-10
105	105	16.0	105	91	11
106-108		16.1-TP	106-107		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	3.2-11.2									
	11.3-15.6 [2]	4-17.0		12-12.1		2.1-4.4.4				
105	16.0 [2]	18.0	all	64	13.4	105	104	13.1	13.18	2.5
	16.1 [2]									

Notes | Test on a real browser | Known issues (0) | Resources (8) | Feedback

WebKit status: Partially Supported

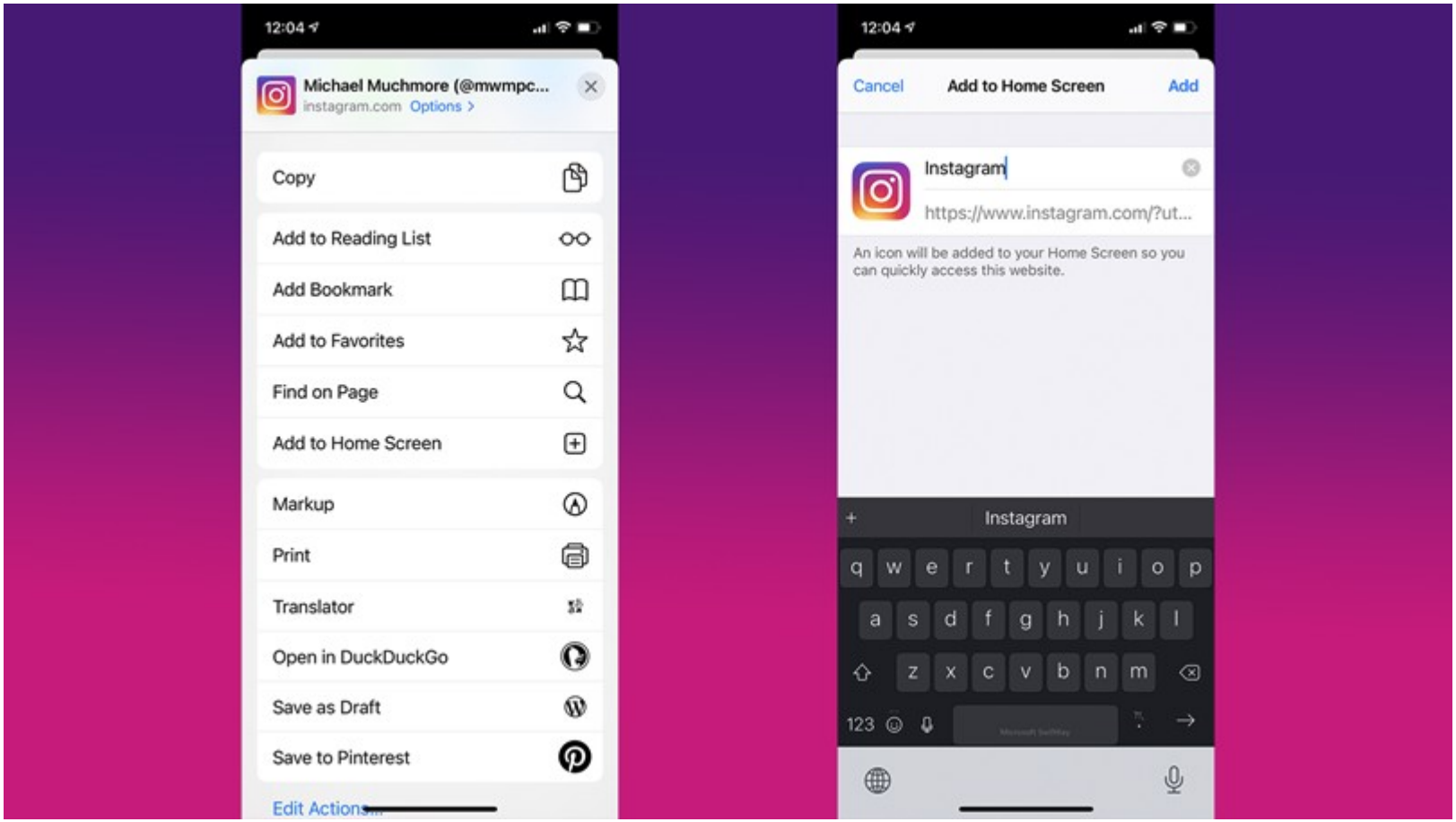
[1] A manifest could be used to list apps in the Microsoft Store, which would use Edge as a back-end.

[2] Safari on iOS does not support A2HS in WebViews like Chrome and Firefox.

[3] Firefox is experimenting with desktop support behind the `browser.ssb.enabled` flag.

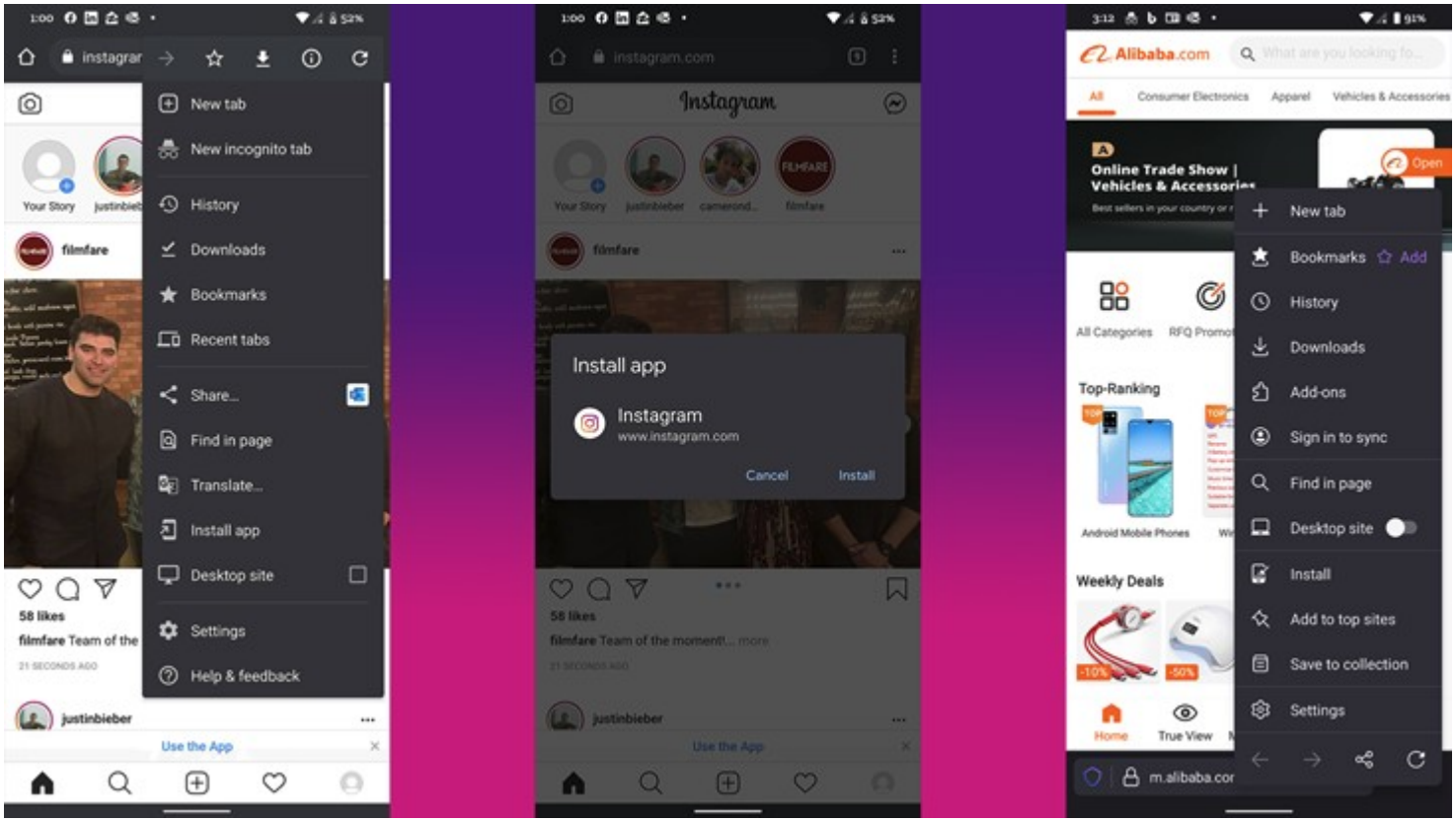
https://caniuse.com/?search=a2hs

Safari on iOS and iPadOS



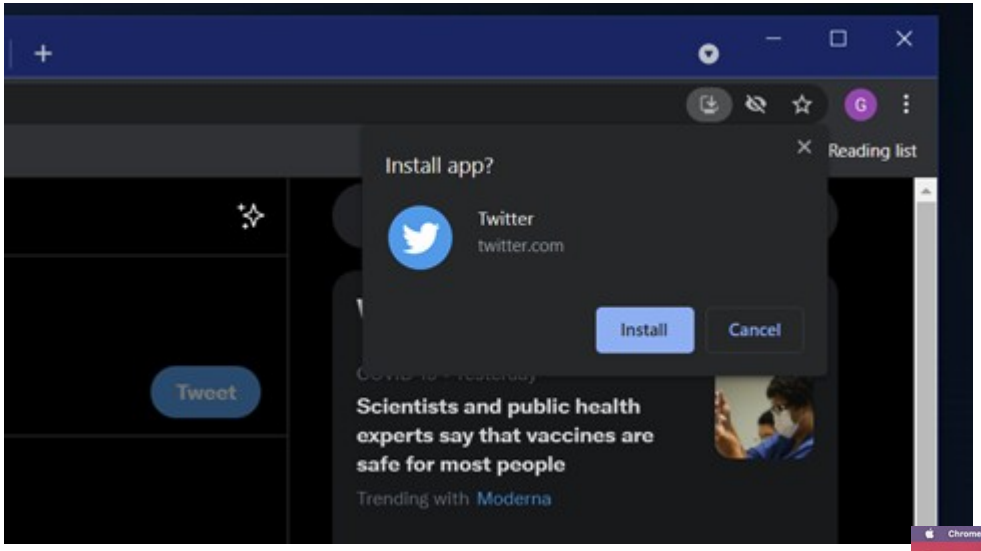
https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how

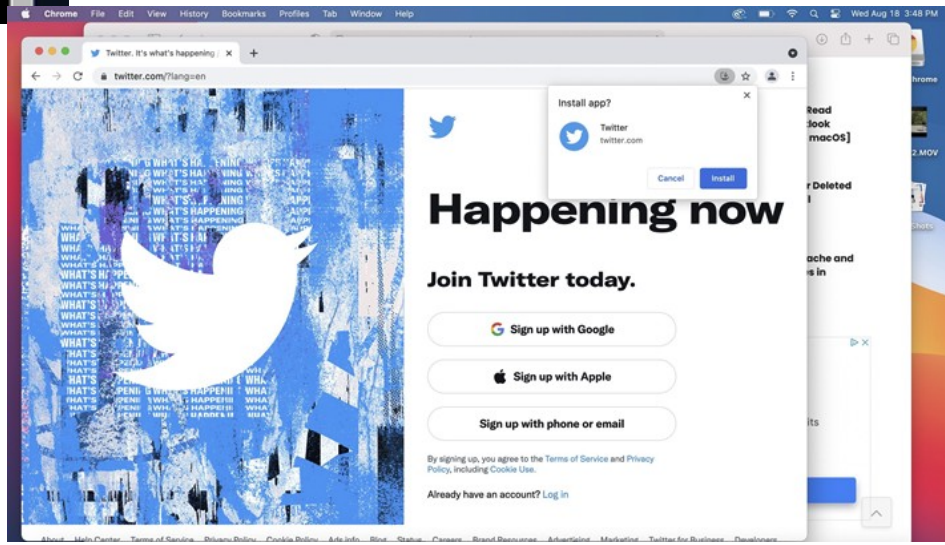
Chrome(left) and Firefox(right) on Android



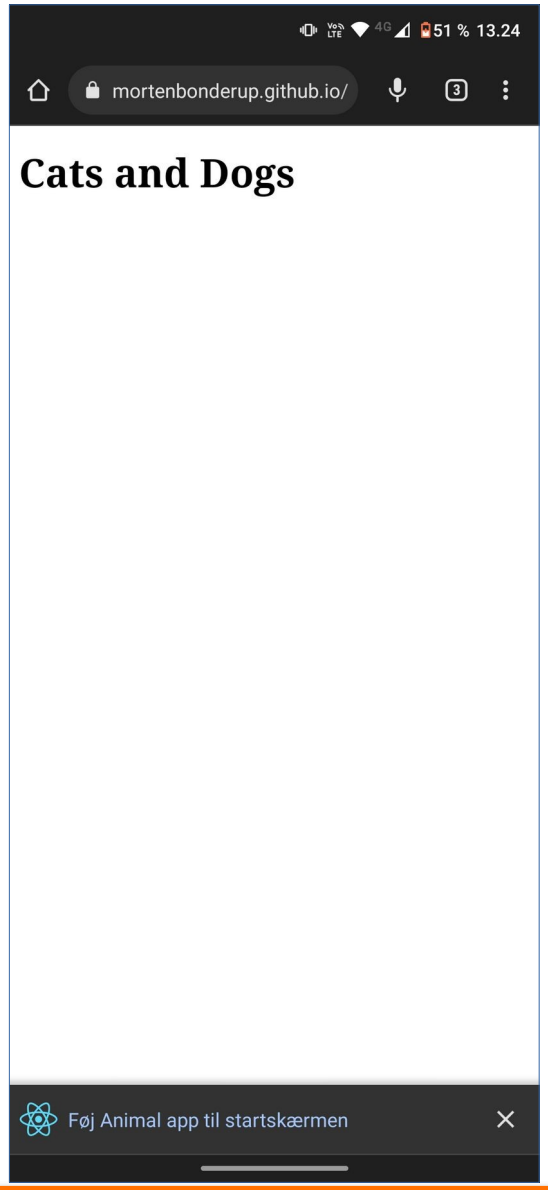
https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how

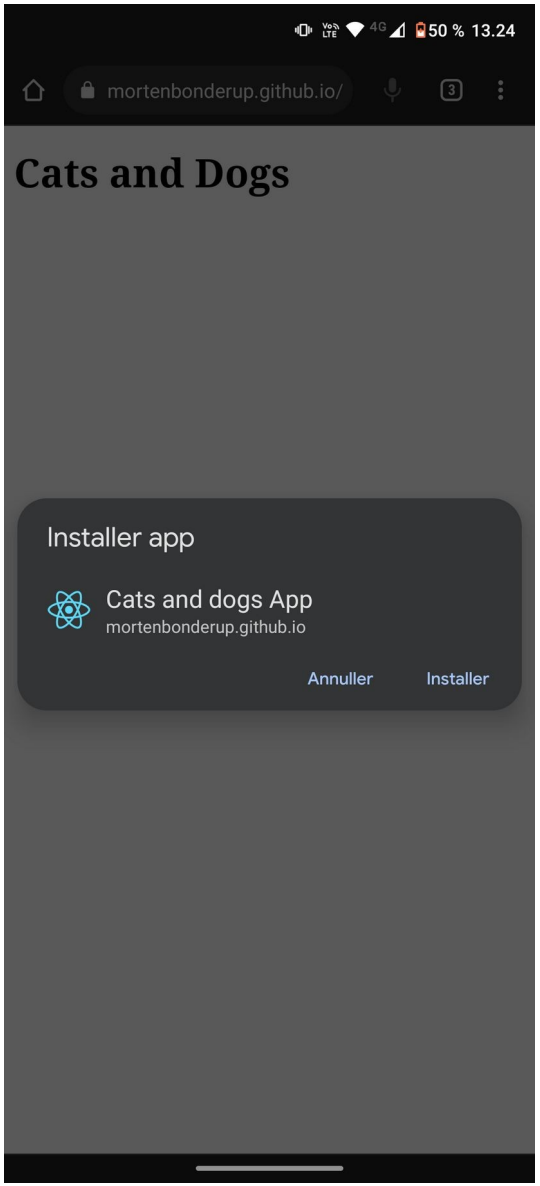
Chrome(left) on Windows and on MacOS(right)





https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how





Mortens android phone

What does it take, if a web app wants to be installable?

1) A web app manifest file with specific keys/values

2) An icon to represent the app on the device

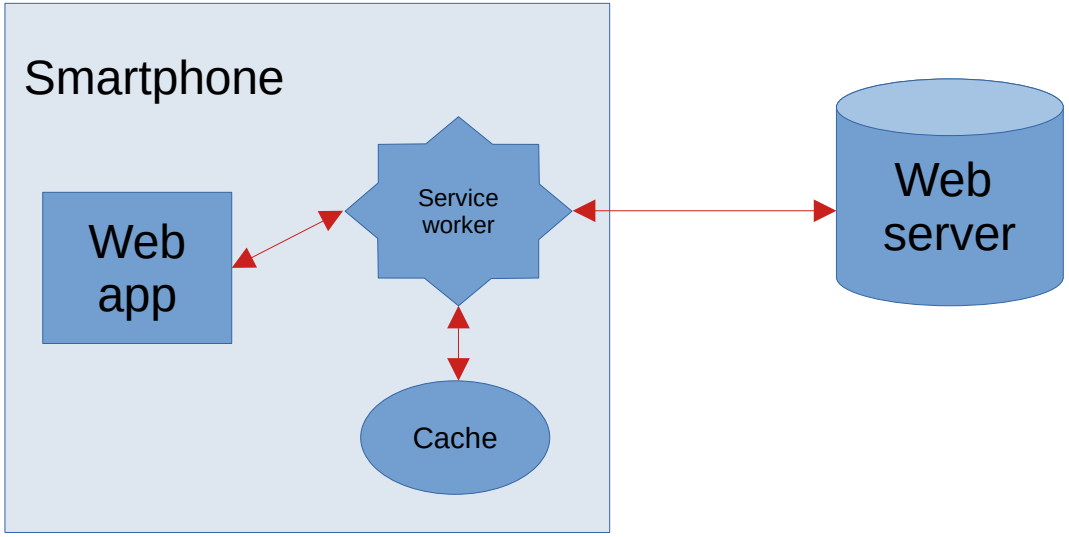
3) A javascript service worker registration to allow the web app to work offline

4) A HTTPS connection

Look here for more details about manifest keys/values:
https://developer.mozilla.org/en-US/docs/Web/Manifest

Service workers

Service Workers are a virtual proxy between the browser and the network. They finally fix issues that front-end developers have struggled with for years — most notably how to properly cache the assets of a website and make them available when the user's device is offline.



https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

Let us check if your browser(s) supports serviceworker

This code checks if the browser supports service workers.

```
if ('serviceWorker' in navigator) {
  // Supported!
}
```

Let us make a project folder and an index.html

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/serviceWorker

What does it take, if a web app wants to be installable?

1) A web app manifest file with specific keys/values

2) An icon to represent the app on the device

3) A javascript service worker registration to allow the web app to work offline

4) A HTTPS connection

Look here for more details about manifest keys/values:
https://developer.mozilla.org/en-US/docs/Web/Manifest

The manifest is a json file

```
{
  "name": "Add to Homescreen App",
  "short_name": "Homescreen App",
  "description": "This app demonstrates the use of Add To Homescreen.",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#d676ff",
  "icons": [
    {
      "src": "logo-192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "/test/index.html"
}
```

```json
{
  "name": "Add to Homescreen App",
  "short_name": "Homescreen App",
  "description": "This app demonstrates the use of Add To Homescreen.",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#d676ff",
  "icons": [
    {
      "src": "logo-192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "/test/index.html"
}
```

- Name: Application name displayed to the user
- Short_name: Displayed to the user if no room for ”Name”
- Description: Developer description of the the app does
- Display: Preferred display mode (fullscreen, standalone, minimal-ui, browser)
- Background_color: Background color for splash screen (if supported)
- Theme_color: Defines the theme color (can vary from OS to OS).
- Icons: A list with available icons.
- Start_url: A relative or absolute URL to the main landing page

https://developer.mozilla.org/en-US/docs/Web/Manifest

Let us make a manifest.json document

1) Create a manifest.json file
2) It should have this content

```
{
  "name": "Cats and Dogs App",
  "short_name": "Animal App",
  "description": "This app loves cats and dogs.",
  "display": "standalone",
  "icons": [
    {
      "src": "icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    },

  ],
  "start_url": "/cat/index.html"
}
```

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/serviceWorker

What does it take, if a web app wants to be installable?

1) A web app manifest file with specific keys/values
2) An icon to represent the app on the device
3) A javascript service worker registration to allow the web app to work offline
4) A HTTPS connection

Look here for more details about manifest keys/values:
https://developer.mozilla.org/en-US/docs/Web/Manifest

Let us get the icons (or you can do one yourself, the app theme is cats and dogs)

1) From todays canvas page, download the icon.zip file and unzip it in your project folder.

What does it take, if a web app wants to be installable?

1) A web app manifest file with specific keys/values
2) An icon to represent the app on the device
3) A javascript service worker registration to allow the web app to work offline
4) A HTTPS connection

Look here for more details about manifest keys/values:
https://developer.mozilla.org/en-US/docs/Web/Manifest

Let’s make som changes to the index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Cats and dogs</title>


    <!-- Requirement: Manifest with specific values -->
    <link rel="manifest" href="manifest.json" />
  </head>


  <body>
    <h1>Cats and Dogs</h1>


    <!-- Requirement: Service Worker with fetch handling -->
    <script>
      if ('serviceWorker' in navigator) {
        navigator.serviceWorker.register('/cat/serviceworker.js');
      }
    </script>
  </body>
</html>
```

And now we create a new file called ”serviceworker.js” with the following content:

```
self.addEventListener('fetch', function(event) {
  event.respondWith(fetch(event.request));
});
```

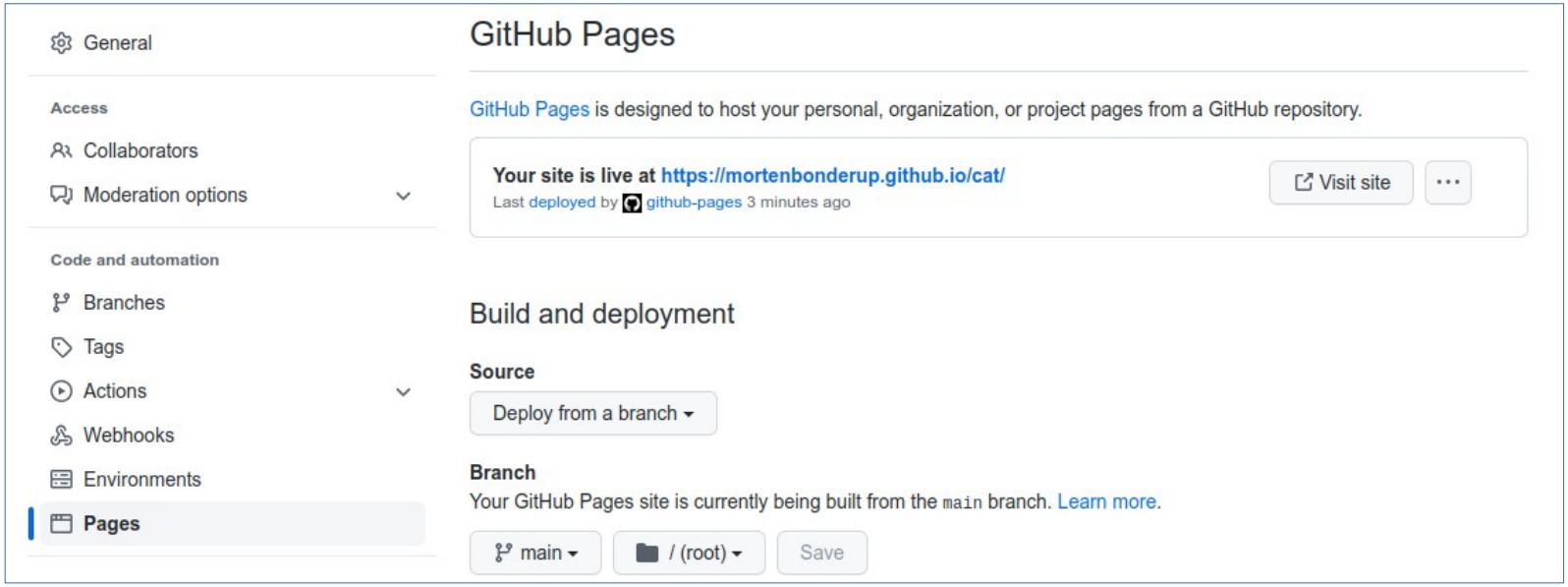
What does it take, if a web app wants to be installable?

1) A web app manifest file with specific keys/values
2) An icon to represent the app on the device
3) A javascript service worker registration to allow the web app to work offline
4) A HTTPS connection

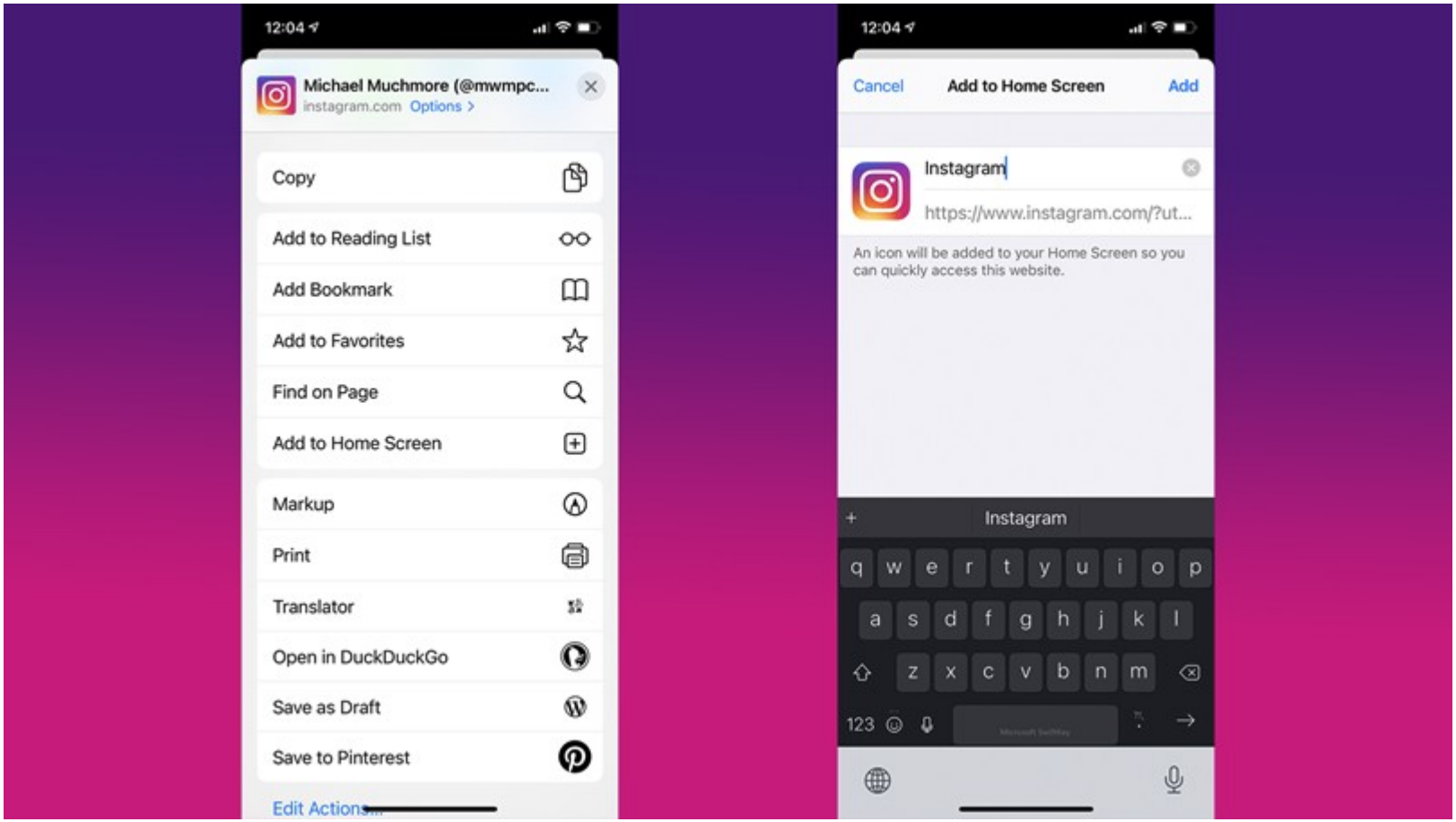
Look here for more details about manifest keys/values:
https://developer.mozilla.org/en-US/docs/Web/Manifest

Let us make a github repo called ”cat” and upload our files to this repo.

1) Upload your files to a github repo called ”cat”.
2) Activate github pages (select branch: main and click on ”save”)
3) Refresh the page after a few minutes to get the url.
4) Try the url on your smartphone (if IOS: Safari) or on your laptop (chrome)
5) If the app installation does not automatically show up, select *install app* in the browser menu (on IOS: add to home screen).

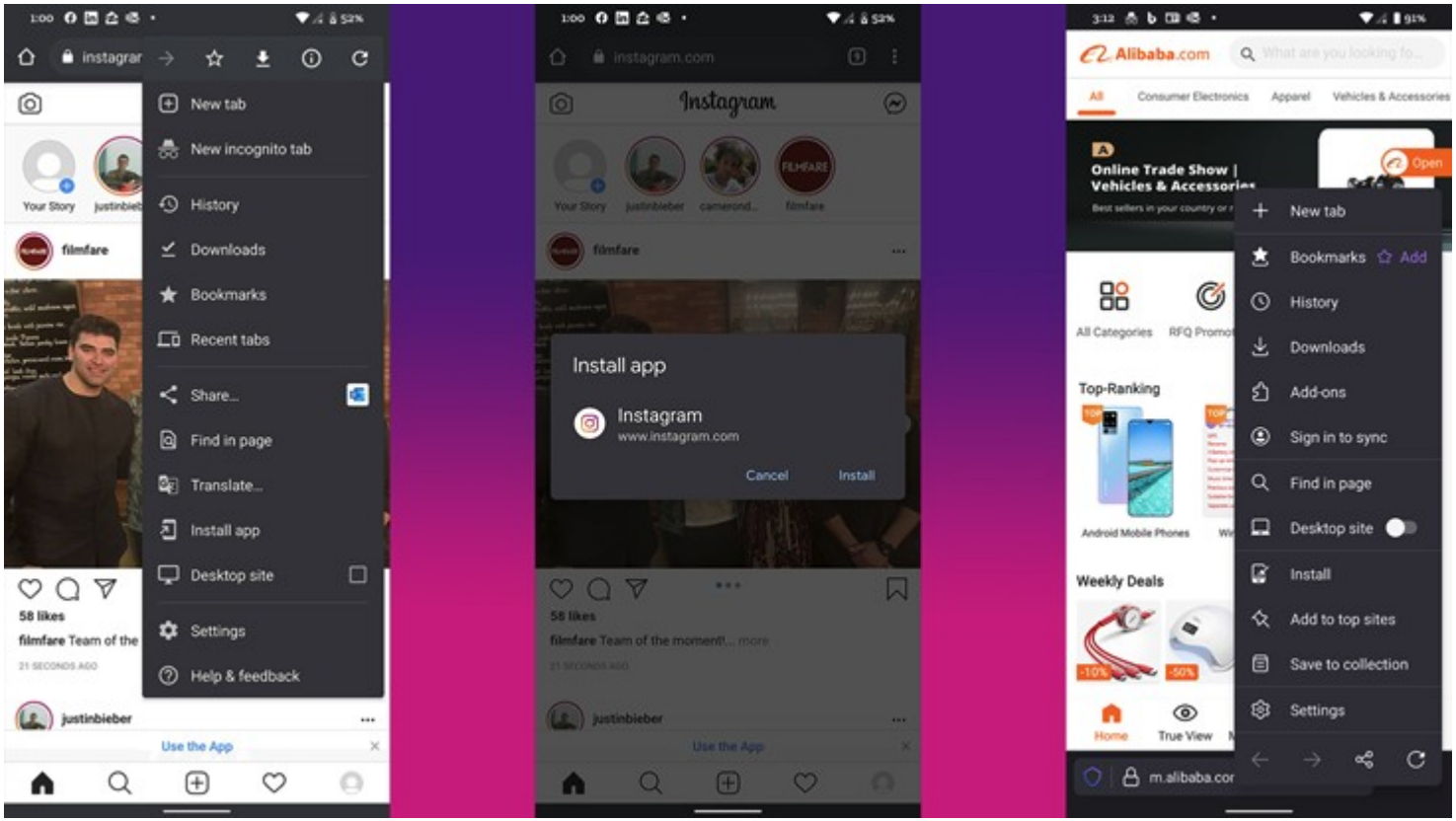


Safari on iOS and iPadOS



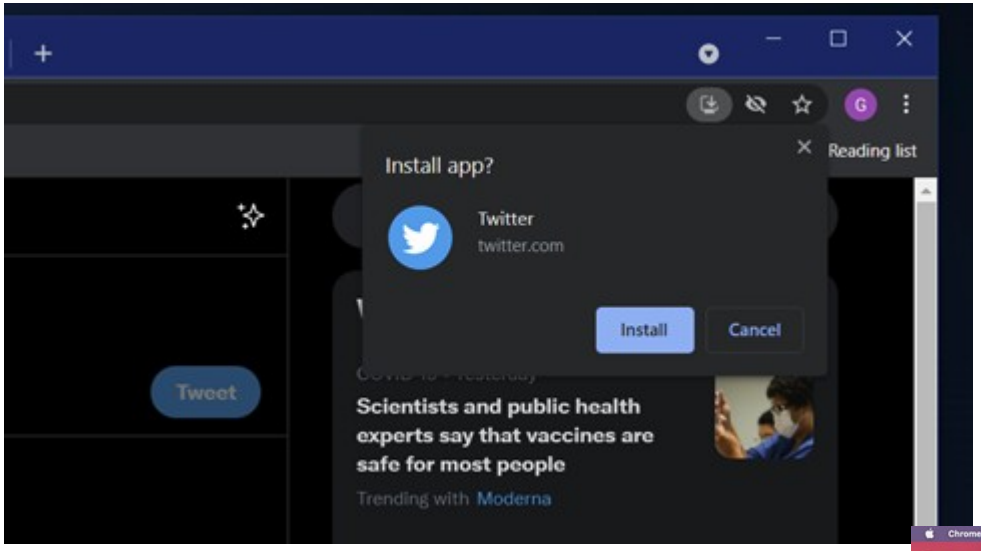
https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how

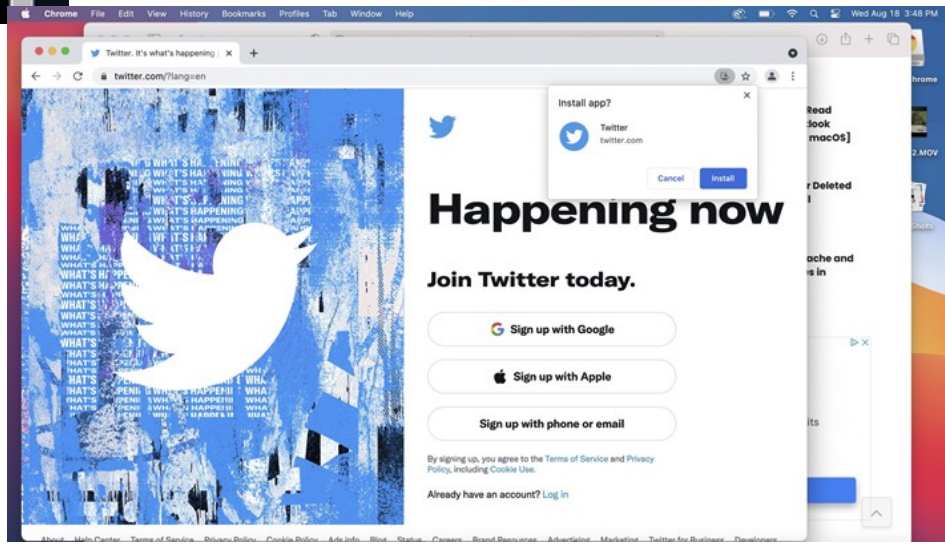
Chrome(left) and Firefox(right) on Android



https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how

Chrome(left) on Windows and on MacOS(right)

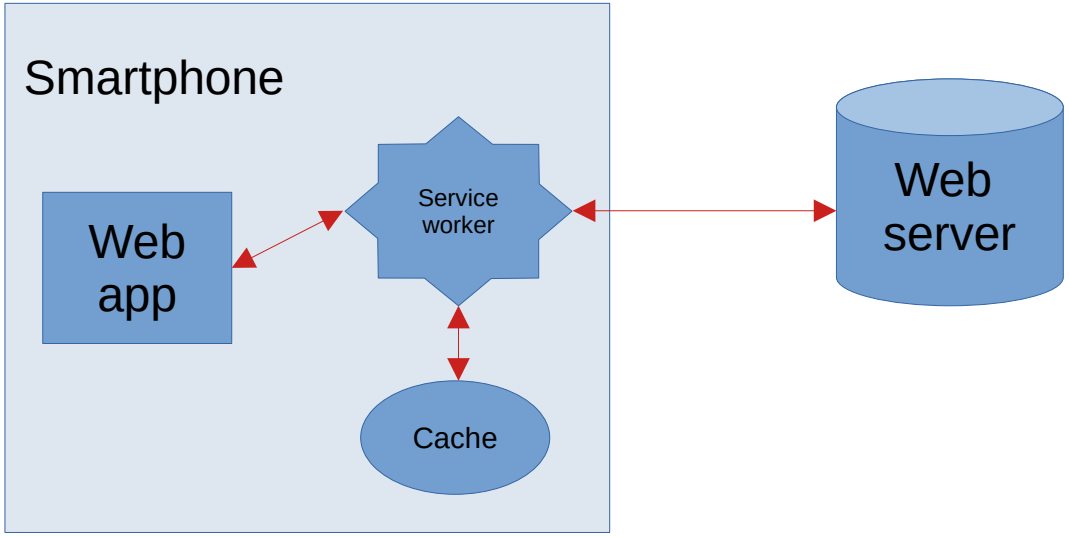




https://uk.pcmag.com/software-services/136092/pwas-turn-websites-into-apps-heres-how

Service workers

Service Workers are a virtual proxy between the browser and the network. They finally fix issues that front-end developers have struggled with for years — most notably how to properly cache the assets of a website and make them available when the user's device is offline.



https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

Caches

- Caches is an asynchronous version of local storage, hence can be used within service workers.
- Caches contains a number of named Cache objects.
- A cache is used to store request and response objects.

Cache Lifetime

- A cache is managed by the author.
  - Creating, updating, expiration and deletion is completely managed by the author, not by the browser.

Let us go offline – in small steps

1) Create a new project folder called ”insects”

2) Make an ”index.html” file with standard content

3) In ”index.html”, register a service worker called ”serviceworker.js”

4) Create an empty ”serviceworker.js” file.

5) Create a manifest file. The app we are to do, is about insects. Use the icons from previous exercise. The app will LATER be located in a github repo called ”insects”.

6) In ”index.html”, link to a manifest.json file.

7) Download ”insects.zip” and ”icons.zip” and unzip it into the project folder.

8) We will code the rest together...

Index.html

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Cache Storage - Insects</title>
  </head>

  <body>
    <h1>Insects</h1>
    <img id="butterflies" src="butterflies.jpg" />
    <img id="butterfly" src="butterfly.jpg" />
    <img id="dragonfly" src="dragonfly.jpg" />

    <script>
      if ('serviceWorker' in navigator) {
        navigator.serviceWorker.register('serviceworker.js');
      }
    </script>
  </body>
</html>
```

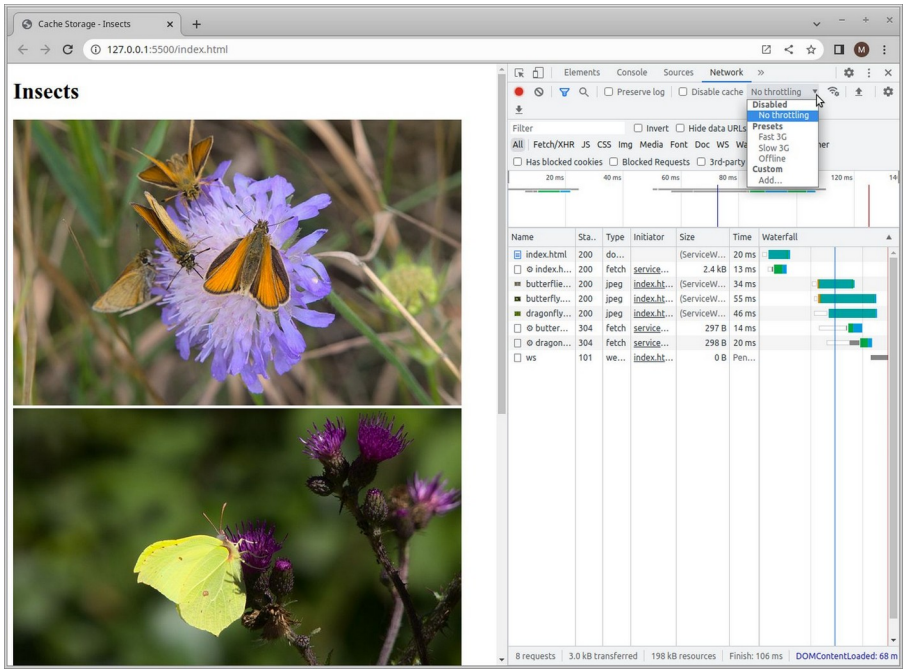
Serviceworker.js

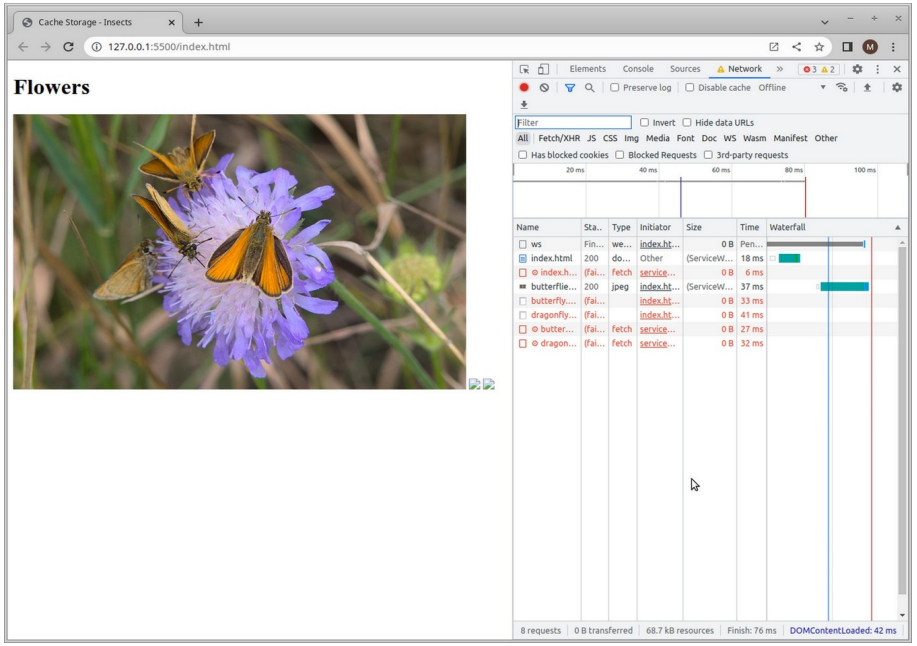
```
const cacheName = 'cache-insects';

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(cacheName).then(function(cache) {
      return cache.addAll(['/', 'index.html', 'butterflies.jpg']);
    })
  );
});

self.addEventListener('fetch', function(event) {
  event.respondWith(
    fetch(event.request).catch(() =>
      caches.open(cacheName).then(cache => cache.match(event.request))
    )
  );
});
```

Testing offline/online capabilitites

In chrome browser

In the developer tools → network tab → No throttling / offline

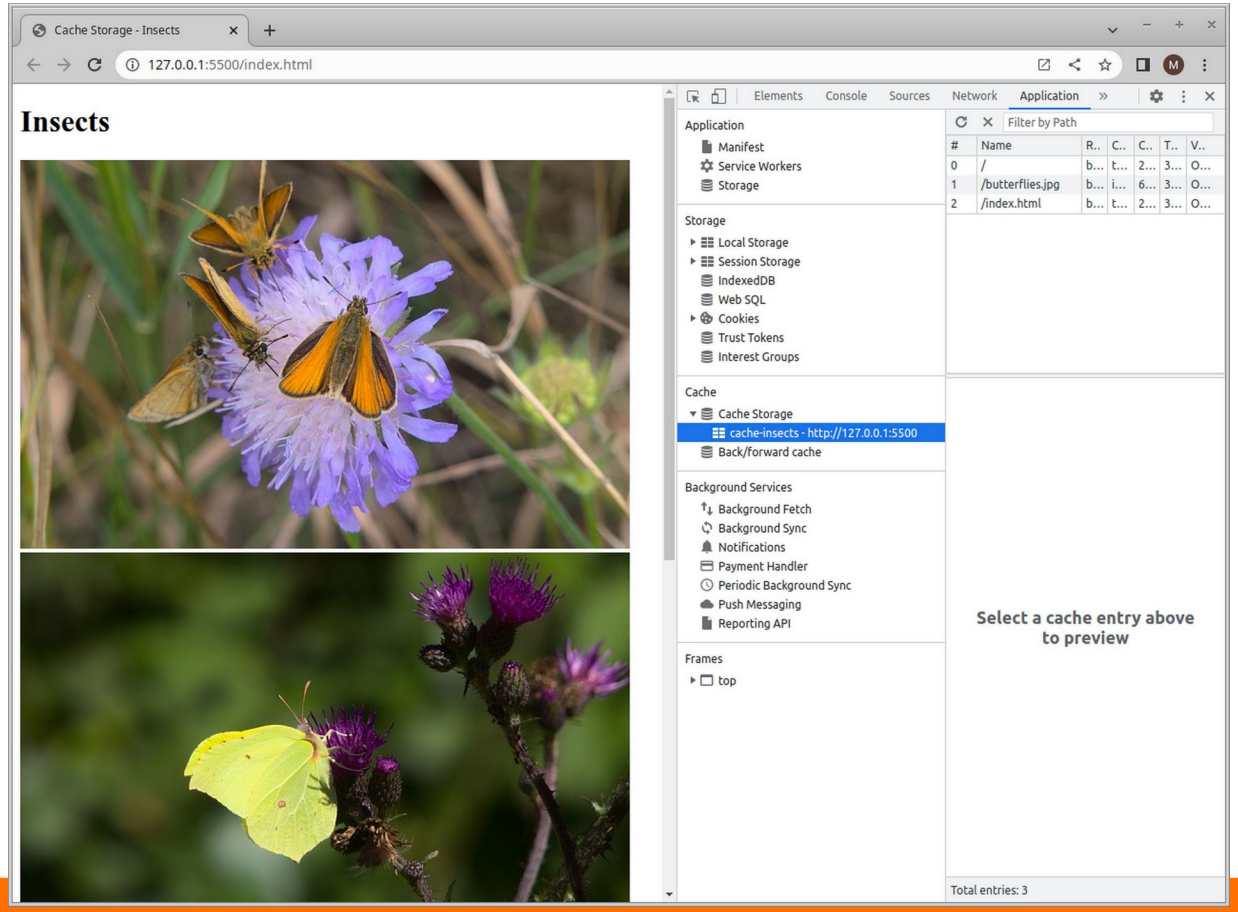




Inspecting the cache

In chrome browser

In the developer tools → application tab → cache



Let us upload it to github and test it on smartphone

1) Add ”/insects/” in front of filenames in JavaScript parts/documents

2) Create a github repo called ”insects” and upload your files

3) Install it on your mobile and test it. Test online/offline capabilities – turning connection on/off (set your mobile in flightmode).

4) It can be quite hard to test – open/close the app several times – at one point in time, butterfly and dragonfly images will disappear when your device is offline because they are not cached.

5) Update the serviceworker to also cache the ”butterfly.jpg” and the ”dragonfly.jpg”.

6) Test that everything is available offline.

Assignment

1) Return to ”Cats and Dogs”
2) On pixabay – find images of cats and/or dogs
3) Insert these images in the index.html
4) Update the service worker, so it caches all files
5) Update the githup repo
6) Delete the app from your phone
7) Visit github cats and dogs and install the app again
8) Test it! On mobile, on desktop.

Assignment

1) Download and unzip ”json_student.zip” from todays Canvas folder
2) Make the solution installable from Github
3) Test it on your mobile and desktop. Online / offline.
4) Add a member to the json file and update Github
5) Does the app recognize the update ?