SENG 365 Week 11

Web Storage and Progressive Web Apps

Assignment 2

Submission requirements

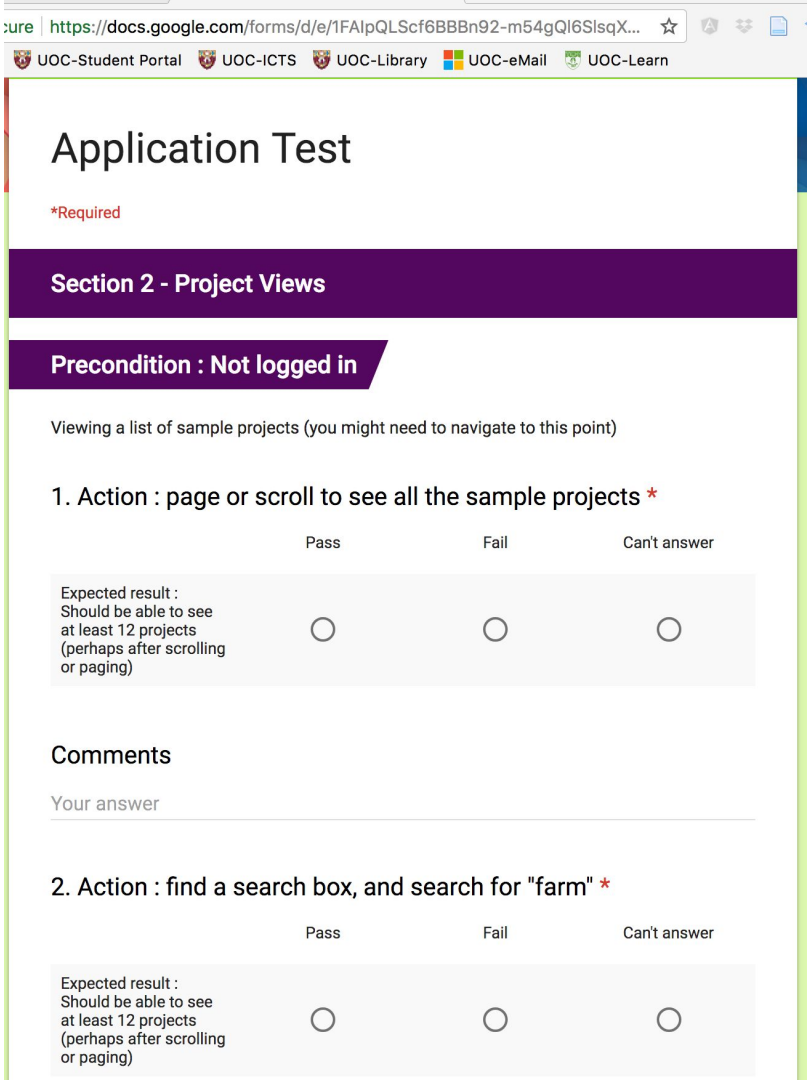
- Zipped project
- Use username as zip filename e.g., <usercode>.zip
- Without `node_modules` to reduce size
- No server code

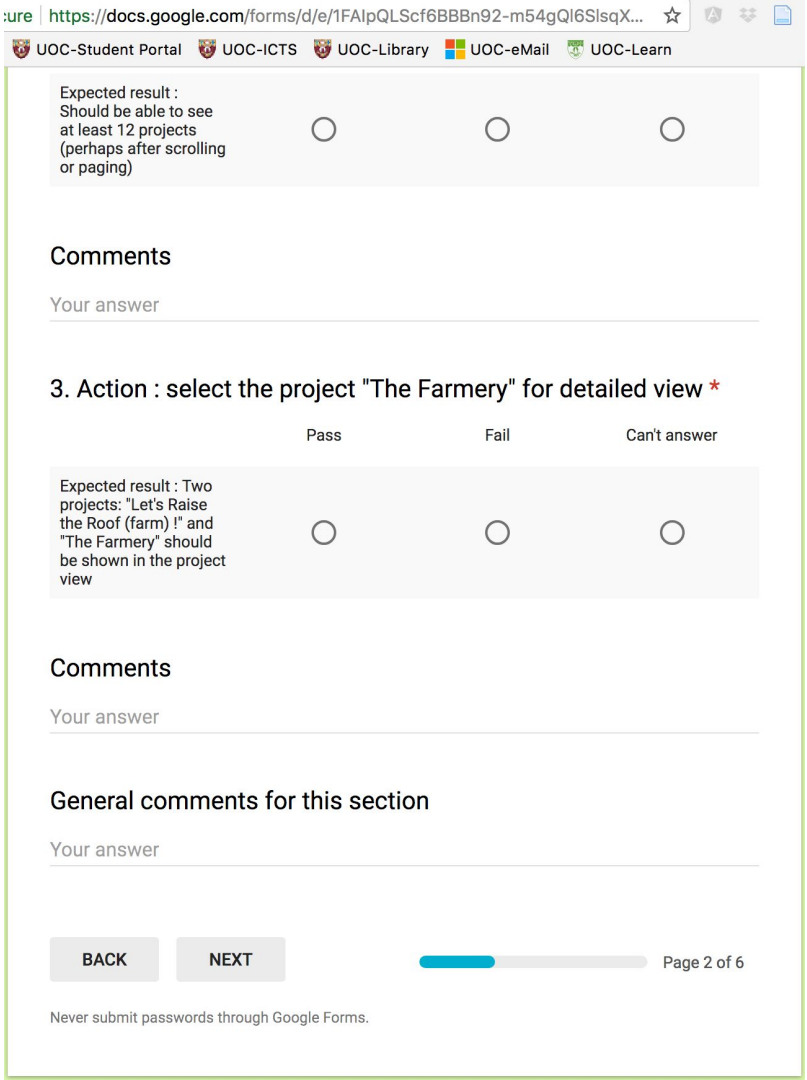
During the test

- Preparation
  - Download .zip of apps
  - Download latest version of server, install it and run it
- For *each* application
  - Reset the database
    - First /reset and /resample database using Postman
  - Install and run the client-side app
    - npm install and run
    - Open Chrome: localhost:<port>
- Run tests

Fragment from 2017's test script

1. Can the application be run?		YES	NO
2. Project views			
Precondition: not logged in, viewing a list of sample projects (you might need to navigate to get to this point)			
Steps	**Expected**	**Pass**	**Fail**
1. Page or scroll to see all the sample projects	Should be able to see at least 12 projects (perhaps after scrolling or paging)		
2. Find a search box, and search for `farm`	Two projects: `Let's Raise the Roof (Farm)!` and `The Farmery` should be shown in the project view		
3. Select project `The Farmery` for detailed view	Extra information for project should be shown including rewards, progress towards goal, backers and pledges		
	Totals		
Comments			


https://docs.google.com/forms/d/e/1FAIpQLScf6BBBn92-m54gQl6SlsqX...
UOC-Student Portal
UOC-ICTS
UOC-Library
UOC-eMail
UOC-Learn
Application Test
*Required
Section 2 - Project Views
Precondition : Not logged in
Viewing a list of sample projects (you might need to navigate to this point)
1. Action : page or scroll to see all the sample projects *
Pass
Fail
Can't answer
Expected result : Should be able to see at least 12 projects (perhaps after scrolling or paging)
Comments
Your answer
2. Action : find a search box, and search for "farm" *
Pass
Fail
Can't answer
Expected result : Should be able to see at least 12 projects (perhaps after scrolling or paging)


https://docs.google.com/forms/d/e/1FAIpQLScf6BBBn92-m54gQl6SlsqX...
UOC-Student Portal
UOC-ICTS
UOC-Library
UOC-eMail
UOC-Learn
Expected result : Should be able to see at least 12 projects (perhaps after scrolling or paging)
Comments
Your answer
3. Action : select the project "The Farmery" for detailed view *
Pass
Fail
Can't answer
Expected result : Two projects: "Let's Raise the Roof (farm) !" and "The Farmery" should be shown in the project view
Comments
Your answer
General comments for this section
Your answer
BACK
NEXT
Page 2 of 6
Never submit passwords through Google Forms.

This week

- Web storage
- IndexedDB
- CacheStorage
- Progressive Web Apps
- Service Workers
- Web Assembly

Web Storage

Session Storage, Local Storage, IndexedDB

HTML5 Web Storage

- Cookies have limitations
  - Size limited to approx. 4KB
  - Send to the server with each request
- HTML5 web storage
  - Allows up to approx. 5MB (depends on browser implementation)
  - Two types:
    - Local storage
    - Session storage
- Not more secure than cookies, however

Local Storage

- Stores ***permanent*** data for your site (i.e., no expiration date)
- Stores data in **key, value pairs**
- Keys and values are **Strings**
- **Setter** and **getter** functions:

```
localStorage.setItem(key, value)
localStorage.getItem(key)
```

Local Storage example

```
<script>
// Check if the localStorage object exists
if(localStorage) {
    // Store data
    localStorage.setItem("first_name", "Peter");

    // Retrieve data
    alert("Hi, " + localStorage.getItem("first_name"));
} else {
    alert("Sorry, your browser do not support local storage.");
}
</script>
```

Local Storage + Zustand

```
1   const useStore = create((set) => ({
2       collection: getLocalStorage("collection") || [],
3       setCollection: (collection) =>
4           set((state) => {
5               setLocalStorage("collection", collection);
6               return { collection };
7           })
8   }));
9
10  const getLocalStorage = (key) => JSON.parse(window.localStorage.getItem(key));
11  const setLocalStorage = (key, value) =>
12      window.localStorage.setItem(key, JSON.stringify(value));
```

Read `collection` from local storage when the Zustand store is created, or default to []

When `collection` is set in Zustand, also update local storage

Wrappers for Local Storage getters and setters

Session Storage

- Stores ***temporary*** data for your site
- **Deleted** when the session ends (browser or tab is closed by user)
- **Same getters and setters** as Local storage

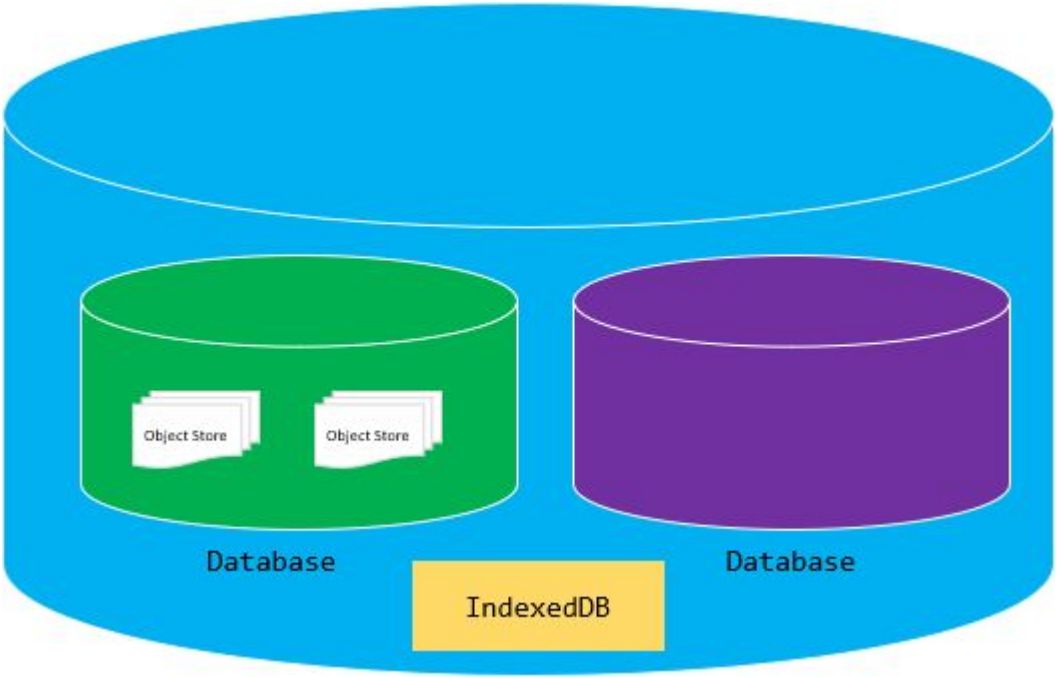
```
<script>
// Check if the sessionStorage object exists
if(sessionStorage) {
    // Store data
    sessionStorage.setItem("last_name", "Parker");

    // Retrieve data
    alert("Hi, " + localStorage.getItem("first_name") + " " +
sessionStorage.getItem("last_name"));
} else {
    alert("Sorry, your browser do not support session storage.");
}
</script>
```

IndexedDB

- Web API for creating indexed NoSQL databases in browser for a web page
- Can create multiple object stores
- Primary keys
- Indexes
- CRUD requests are asynchronous using promises
- Follows same-origin policy (see CORS)

See: https://developers.google.com/web/ilt/pwa/working-with-indexeddb

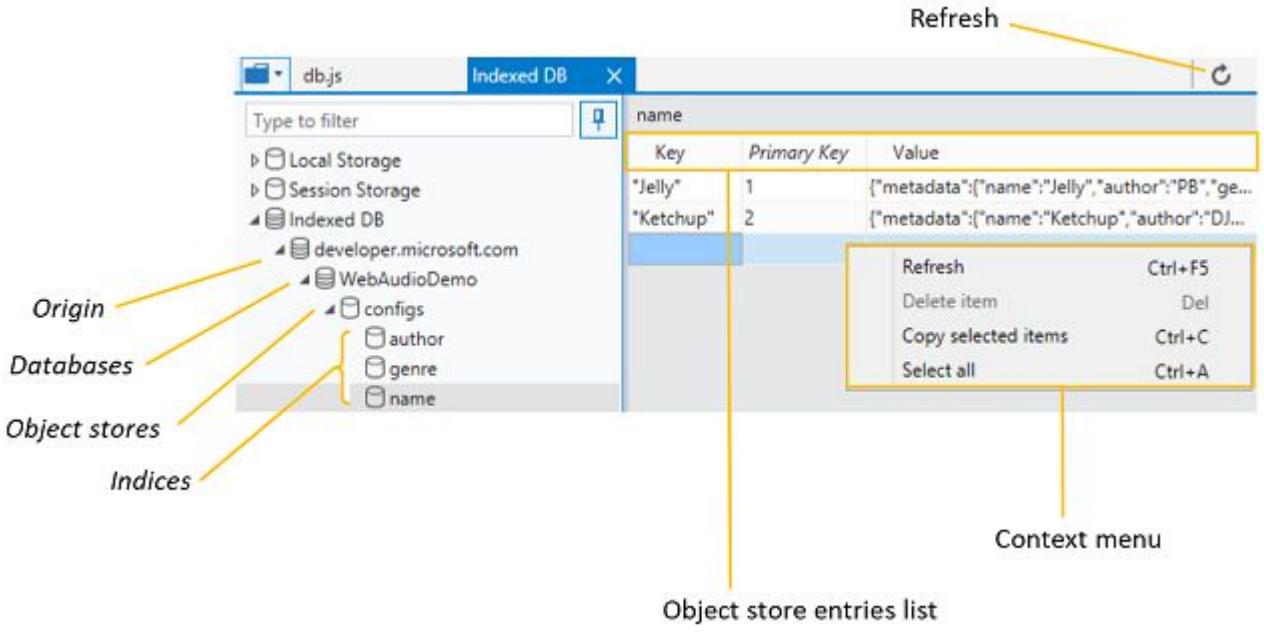
IndexedDB



One or more databases

Each database made up of one or more Object Stores

IndexedDB elements



IndexedDB Object Store

- Conceptually similar to database table, but NoSQL
- Records – key, value pairs
- Create indexes on object stores
- Read/write is transactional

`indexedDB.open` callback functions

1. `onerror` - handles an error when opening database
2. `onsuccess` - should execute your transactions in this callback
3. `onupgradeneeded` - executed when a new database is created
    a. Create Object Stores here
    b. “New database” includes a new version number

IndexedDB opening a database

```
if (!window.indexedDB) {
    console.log(`Your browser doesn't support IndexedDB`);
    return;
}

const request = indexedDB.open('contacts-db', 1);
```

Check if the browser supports IndexedDB

Note: `indexedDB.open` returns a Promise.
Can use `await` syntax here.

IndexedDB onerror, onsuccess

```
request.onerror = (event) => {
    console.error(`Database error: ${event.target.errorCode}`);
}

request.onsuccess = (event) => {
    const db = event.target.result;

    // do db transactions here
}
```

IndexedDB onupgradeneeded

```
request.onupgradeneeded = (event) => {
    let db = event.target.result;

    // create the Contacts object store
    // with auto-increment id
    let store = db.createObjectStore('Contacts', {
        autoIncrement: true
    });

    // create an index on the email property
    let index = store.createIndex('email', 'email', {
        unique: true
    });
}
```

Create a new object store called Contacts and give it an auto incrementing key

Create an index on the email field and make sure that each email is unique. It will throw an error if you attempt to add 2 records with the same email.

IndexedDB onsuccess populating the database

```
request.onsuccess = (event) => {
    const db = event.target.result;

    insertContact(db, {
        email: 'john.doe@outlook.com',
        firstName: 'John',
        lastName: 'Doe'
    });

    insertContact(db, {
        email: 'jane.doe@gmail.com',
        firstName: 'Jane',
        lastName: 'Doe'
    })
}
```

```
function insertContact(db, contact) {
    // create a new transaction
    const txn = db.transaction('Contacts', 'readwrite');

    // get the Contacts object store
    const store = txn.objectStore('Contacts');
    let query = store.put(contact);

    //handle success case
    query.onsuccess = (event) => {
        console.log(event);
    }

    // handle the error case
    query.onerror = (event) => {
        console.log(event.target.error);
    }

    // close the database once the transaction completes
    txn.oncomplete = () => {
        db.close();
    };
}
```

Cache Storage API

- Stores pairs of Request and Response objects
- Cache can be hundreds of megabytes in size
- Access cache:
```
const cache = await caches.open('my-cache');
```
- Adding to cache
  - add –
```
cache.add(new Request('/data.json'));
```
  - addAll –
```
const urls = ['/weather/today.json', '/weather/tomorrow.json'];
cache.addAll(urls);
```
  - put –
```
cache.put('/test.json', new Response('{"foo": "bar"}'));
```
- Retrieving from cache
```
const response = await cache.match(request);
console.log(request, response);
```

Progressive Web Apps

Progressive web apps

- Web applications designed to appear to be ‘installed’ as native applications
- Begin life in a browser tab...
- ... then can be ‘installed’ as native apps
- Rely on Service Workers
  - Notifications
  - Background sync (offline cache)

For more:
https://developers.google.com/web/progressive-web-apps/checklist

Service Workers

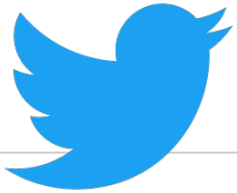
- Proxy ‘servers’ that sit between web applications, and the browser and network
- JavaScript that:
  - runs on its own thread: not blocking
  - is headless (no access to DOM)
- Rely on HTTPS, for security
- Associated with specific server/website

For more:
https://developers.google.com/web/fundamentals/primers/service-workers

Devices running SPAs

- Many form factors and connection types
- All have browsers, but also apps
- One option is to develop web sites and native apps in parallel
- Alternative is progressive web apps

Twitter PWA



- On mobile originally had separate website for in browser and an installable native app from app store.
- Migrated to progressive web app on mobile in 2020
- Advantages cited:
  - Much smaller size
  - Adaptable (no need for app store approvals when making changes)
  - Automatic updates
  - New operating systems
  - Faster, more efficient development

Criteria for Progressive

- Responsive
- Connectivity independent
- App-like interactions
- Fresh
- Safe
- Discoverable
- Re-engagable
- Installable
- Linkable

https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

PWA “Good to haves”

- Mobile-friendly design
- Near-instant loading
  - Interactive in less than 5 sec before Service Worker installed
  - Once Service Worker installed should load in < 2 sec
- Work across devices & browsers
  - 90%+ of all users in market
- Fluid animations
  - Visual transitions

https://infrequently.org/2016/09/what-exactly-makes-something-a-progressive-web-app/

Technical definition of a PWA

- Originate from a **Secure Origin**
- **Load** while **offline**
- Reference a **Web App Manifest**
  - W3C spec defining a JSON-based manifest
  - Web app manifests | MDN
  - `name`
  - `short_name`
  - `start_url`
  - `display`
  - Icon – at least 144x144 px in PNG format

https://infrequently.org/2016/09/what-exactly-makes-something-a-progressive-web-app/

Example manifest

Deployed in HTML using a link tag

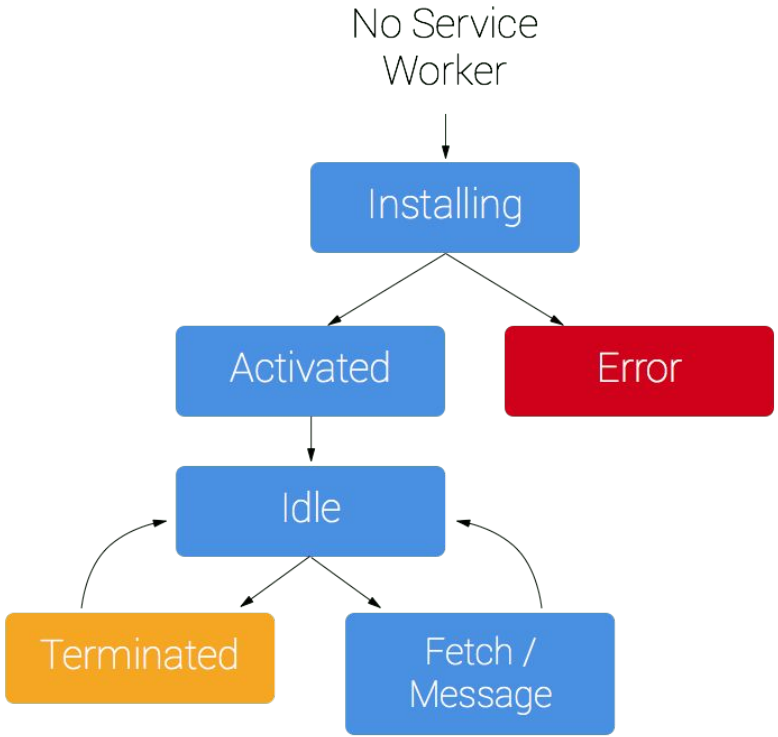
```
<link rel="manifest" href="/manifest.json">
```

```
{
  "short_name": "Weather",
  "name": "Weather: Do I need an umbrella?",
  "icons": [
    {
      "src": "/images/icons-vector.svg",
      "type": "image/svg+xml",
      "sizes": "512x512"
    },
    {
      "src": "/images/icons-192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/images/icons-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/?source=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/",
  "theme_color": "#3367D6",
  "shortcuts": [
    {
      "name": "How's weather today?",
      "short_name": "Today",
      "description": "View weather information for today",
      "url": "/today?source=pwa",
      "icons": [{ "src": "/images/today.png", "sizes": "192x192" }]
    },
    {
      "name": "How's weather tomorrow?",
      "short_name": "Tomorrow",
      "description": "View weather information for tomorrow",
      "url": "/tomorrow?source=pwa",
      "icons": [{ "src": "/images/tomorrow.png", "sizes": "192x192" }]
    }
  ],
  "description": "Weather forecast information",
  "screenshots": [
    {
      "src": "/images/screenshot1.png",
      "type": "image/png",
      "sizes": "540x720"
    },
    {
      "src": "/images/screenshot2.jpg",
      "type": "image/jpg",
      "sizes": "540x720"
    }
  ]
```

Service Worker

- Javascript that runs in the background
- Can be used to execute long-running processes
- Must be started/registered by a web page
- Allows websites to run offline by serving cached data
- Must be served over `https`

Service Worker lifecycle



Registering a Service Worker

- `serviceWorker.register` registers the service worker (js file) for this site
- `scope` defines subset of content that Service Worker controls
- Max scope is the location of the worker

```
const registerServiceWorker = async () => {
  if ('serviceWorker' in navigator) {
    try {
      const registration = await navigator.serviceWorker.register(
        '/sw-test/sw.js',
        {
          scope: '/sw-test/',
        }
      );
      if (registration.installing) {
        console.log('Service worker installing');
      } else if (registration.waiting) {
        console.log('Service worker installed');
      } else if (registration.active) {
        console.log('Service worker active');
      }
    } catch (error) {
      console.error(`Registration failed with ${error}`);
    }
  }
};

// ...

registerServiceWorker();
```

Worker Install event

- When `install` event is triggered, the worker can cache files in its scope
- `self` refers to the worker

```
const addResourcesToCache = async (resources) => {
  const cache = await caches.open("v1");
  await cache.addAll(resources);
};

self.addEventListener("install", (event) => {
  event.waitUntil(
    addResourcesToCache([
      "/sw-test/",
      "/sw-test/index.html",
      "/sw-test/style.css",
      "/sw-test/app.js",
      "/sw-test/image-list.js",
      "/sw-test/star-wars-logo.jpg",
      "/sw-test/gallery/bountyHunters.jpg",
      "/sw-test/gallery/myLittleVader.jpg",
      "/sw-test/gallery/snowTroopers.jpg",
    ])
  );
});
```

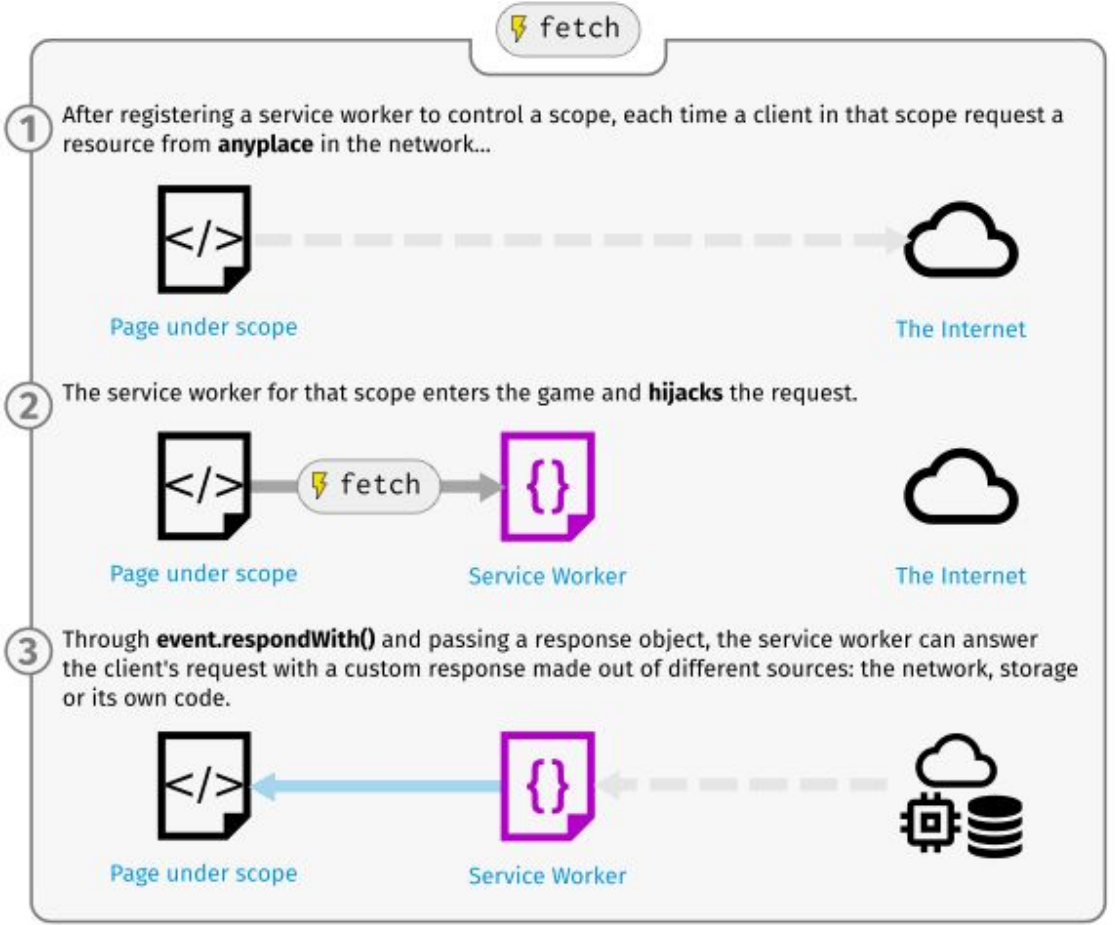
This code is inside the worker: `/sw-test/sw.js`

Serving data when offline

```
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request)
  );
});
```

This example listens for `fetch` and returns the cached data, but you can return anything you want in `event.respondWith`!

If data is not cached, it could make network request and update the cache.



Data storage in a PWA

- Web Storage (Local storage)
- IndexedDB
- Service Workers and cached resources (using CacheStorage API)

WebAssembly

- Binary code that is pre-compiled to WebAssembly (wasm) from other languages, incl.:
  - Emscripten (C, C++)
  - Rust
  - AssemblyScript (TypeScript)
  - TinyGo (Go)
- Ahead-of-time or Just-in-time compilation in browser
- 2019 W3C recommendation
- Can be a supporting technology for PWA, but designed for any high-performance web page

WASM applications

Support for languages and toolkits

Image / video editing.

Games (e.g. with heavy assets)

Peer-to-peer applications

Music applications

Image recognition

VR and augmented reality

CAD applications

Scientific visualization and simulation

Interactive educational software, and news articles

Platform simulation / emulation

Language interpreters and VMs

POSIX user-space environment

Developer tooling

Remote desktop

VPN

Encryption

Local web server

Fat client for enterprise applications (e.g. databases)

WebAssembly bytecode

C source code	WebAssembly `.wat` text format	WebAssembly `.wasm` binary format

`int factorial(int n) {`
`  if (n == 0)`
`  return 1;`
`  else`
`  return n * factorial(n-1);`
`}`
	`(func (param i64) (result i64)`
	`  local.get 0`
	`  i64.eqz`
	`  if (result i64)`
	`  i64.const 1`
	`  else`
	`  local.get 0`
	`  local.get 0`
	`  i64.const 1`
	`  i64.sub`
	`  call 0`
	`  i64.mul`
	`  end)`
		`00 61 73 6D 01 00 00 00`
		`01 00 01 60 01 73 01 73 06`
		`03 00 01 00 02`
		`0A 00 01`
		`00 00`
		`20 00`
		`50`
		`04 7E`
		`42 01`
		`05`
		`20 00`
		`20 00`
		`42 01`
		`7D`
		`10 00`
		`7E`
		`0B`
		`0B 15 17`

More examples: https://wasmbyexample.dev/home.en-us.html

Web Technologies

- Client-side tech is now much more than HTML/JS/CSS