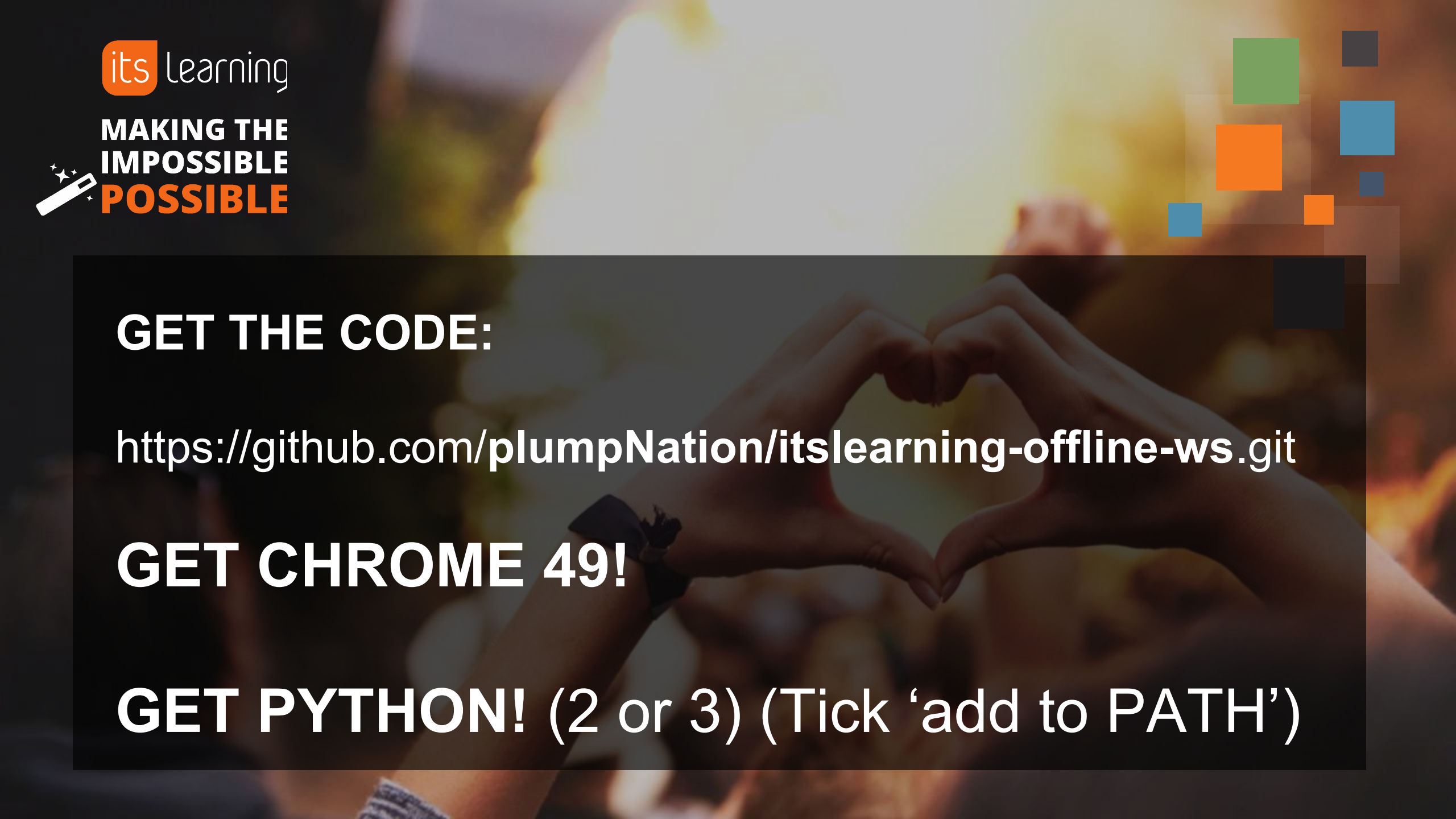

its learning
MAKING THE IMPOSSIBLE POSSIBLE
GET THE CODE:
https://github.com/plumpNation/itslearning-offline-ws.git
GET CHROME 49!
GET PYTHON! (2 or 3) (Tick ‘add to PATH’)

its learning

MAKING THE IMPOSSIBLE POSSIBLE

Offline APIs in the browser

gavin king

developer

Mobile apps have many advantages over a browser application.

One of the biggest advantages they have is that they can **work really well offline.**

Offline is not just offline

It could be:

- when the user has **no connection** to the network
- when **your server is unavailable**

What does an offline experience offer the end user?

Perceived Performance

Data can be cached locally, with the end result being a blisteringly fast page load.

video

Less server traffic

Using the CacheStorage you will guarantee returning the local version immediately, no round trips for etags, nothing.

Background sync

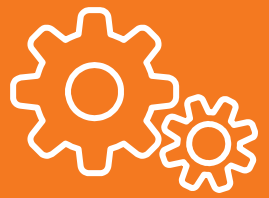
The main application is offline, data syncs with the cloud when a user is online.

Available in Chrome 49

An expected user experience

The web is **starting** to become an application platform. User's have expectations from 'apps'.

What offline techniques?



ServiceWorker API

Install your application locally and intercept requests, choosing how to response.

Offline/Online detection

offline and **online** events and **navigator.onLine** give a rudimentary ability to make decisions about a course of action.

Cache API

Gives fine grained control over asset caching.

Used best in conjunction with Service Workers.

browser storage

IndexedDB, localStorage, sessionStorage; we've heard it all before.

Start the workshop!

- Inside the **workshop** folder, you will find numbered folders which contain a '**start**' and a '**finished**' folder.
- Barebones examples for techniques are included in the '**examples**' folder.
- If you fall behind or start to get frustrated, please just go to the '**finished**' folder and check out the solution there.

0: Offline detection

Offline detection

- The ‘**navigator.onLine**’ property.
  Note: camelcase.

- There are 2 window events you can listen to ‘**offline**’ and ‘**online**’.

navigator.onLine

===

true/false

```
window.addEventListener

'offline'

'online'
```

The exercise

- Open **workshop/0/start/app.js** in your IDE.
- **We'll add code** to the **app.js** script which **detects** whether you are online or offline and fires a callback.
- Add/remove classes "**online**" and "**offline**" to the **<footer id="network-indicator">** in your html page.
- If you get stuck or need to see an example, flick through the **finished** folder.

Some code to look at

```
window.addEventListener('offline', changeNetworkState);
window.addEventListener('online, changeNetworkState);

if (navigator.onLine) {
  // do something
}

let element = document.getElementById('network-
indicator');

// to completely replace the class on an element...
element.className = 'offline';
```


EQUIPMENT PREPARED
LET'S BEGIN
Troll.me

Start the server

```
# run from the root of your repository

$ python -m SimpleHTTPServer 8000

# or for python 3

$ python -m http.server 8000

Serving HTTP on 0.0.0.0 port 8000 ...
```

The exercise

- Open **http://localhost:8000/workshop/1/start**

  OR

  http://localhost:8000/workshop/1/finished

- Turn your network on and off and watch the result.

Section recap

- Learned about the **onLine** property on the navigator object.
- Added detection for offline using the browser offline and online events.
- Updated the DOM based on connection status.

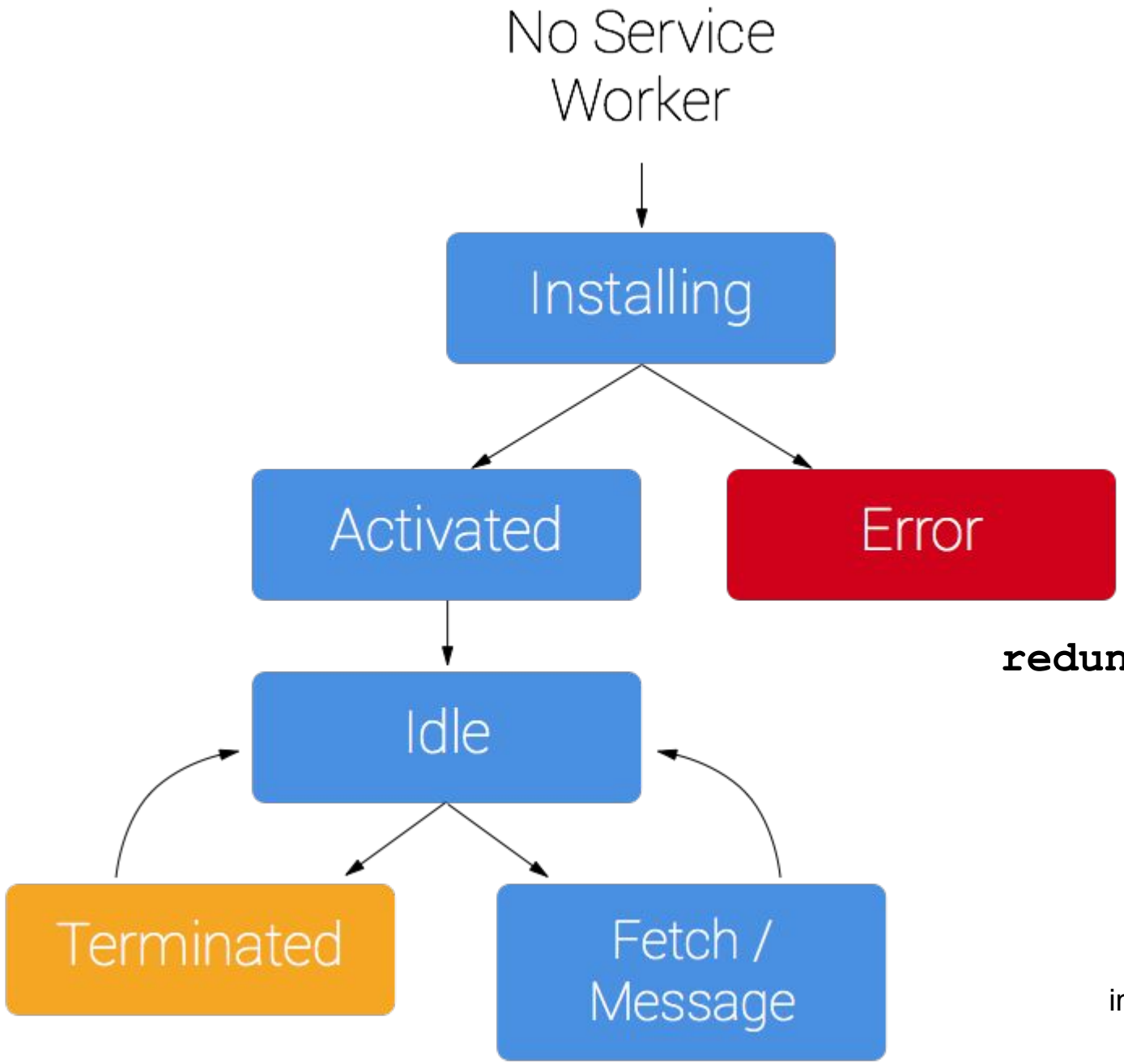
1: ServiceWorker lifecycle

SERVICE WORKERS

take control!

What is a ServiceWorker?

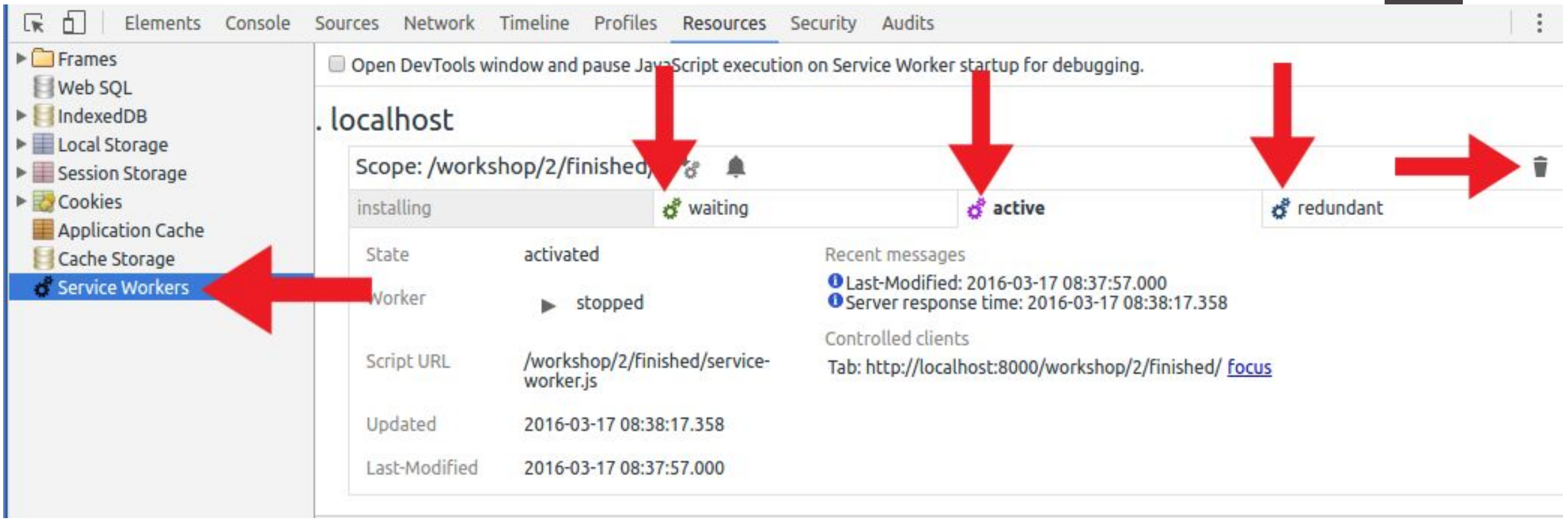
- A service worker is just a javascript file that your browser stores by domain. You can think of it as a **request proxy** that lives locally and runs for a domain even if you are offline.
- It has a lifecycle.
- It's **normally only https**, but Chrome will allow http for localhost..
- It has a **scope**, and cannot reach upward out of that scope to fetch files etc.
  - It's scope defaults to the folder it is in.



`redundant`

image taken from [html5rocks]

tooling


Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
Application Cache
Cache Storage
Service Workers
Open DevTools window and pause JavaScript execution on Service Worker startup for debugging.
. localhost
Scope: /workshop/2/finished/
installing
waiting
active
redundant
State
activated
Worker
stopped
Script URL
/workshop/2/finished/service-worker.js
Updated
2016-03-17 08:38:17.358
Last-Modified
2016-03-17 08:37:57.000
Recent messages
Last-Modified: 2016-03-17 08:37:57.000
Server response time: 2016-03-17 08:38:17.358
Controlled clients
Tab: http://localhost:8000/workshop/2/finished/ focus

The exercise

- Open the **workshop/1/start** folder.

- Create a **service-worker.js** file lifecycle events.
  - ‘install’
  - ‘activate’
  - ‘fetch’

- Register the service worker in app.js
  - use **navigator.serviceWorker.register**

- Open the **Resources** tab in your developer tools and watch the service worker lifecycle in action.

Some code to look at

```
// in app.js
navigator.serviceWorker.register('sw.js');


// in sw.js
self.addEventListener('install', callback);

self.addEventListener('activate', callback);

self.addEventListener('fetch', callback);
```

The exercise (continued)

- Click the **inspect** link in the a service worker tab to access a separate console for an individual service worker.
- To force **waiting**, make changes in the service worker and refresh the page.
- To transition from **waiting** to **active**, close the tab and open it again (or delete the active service worker).
- To force the service worker to be **redundant,** throw an error in your install callback.

Section recap

- Creating and registering a service worker
- Observed the lifecycle of the service worker and states:
  - ❑ installing
  - ❑ waiting
  - ❑ active
  - ❑ redundant

2: ServiceWorker intercept requests

The exercise

- Open the **workshop/2/start** folder.
  (Note that the last exercise has been moved to a helper.)

- You should have a service worker to modify.

- Intercept a request for the news.json
  - Respond with the your own news item.
  - You can hardcode it in your service worker.

The exercise (continued)

- Requests are ‘heard’ when the **‘fetch’** event is emitted in the service worker.
- **‘event.request.url’** contains the path of the request
- if it ‘**endsWith’** ‘news.json’ then you will need to take control.
- the fetch event callback must use ‘**event.respondWith’** to return a ‘**Response’** object, or default browser behaviour will happen.

Some code to look at

```
self.addEventListener('fetch', function (event) {
  // Remember to stringify objects to use in responses
  let newsObject = JSON.stringify({ … });

  if (!event.request.url.endsWith('news.json')) {
    return; // do nothing, and the request will just pass through
  }

  event.respondWith(
    new Response(newsObject)
  );
});
```

Interesting to know

- Observe how many refreshes it takes to use the service worker ‘fetch’ handler.

- Once you see your service worker take control, try using **‘hard refresh’.** (SHIFT + F5)

  Notice, it does **not** fetch through the service worker.

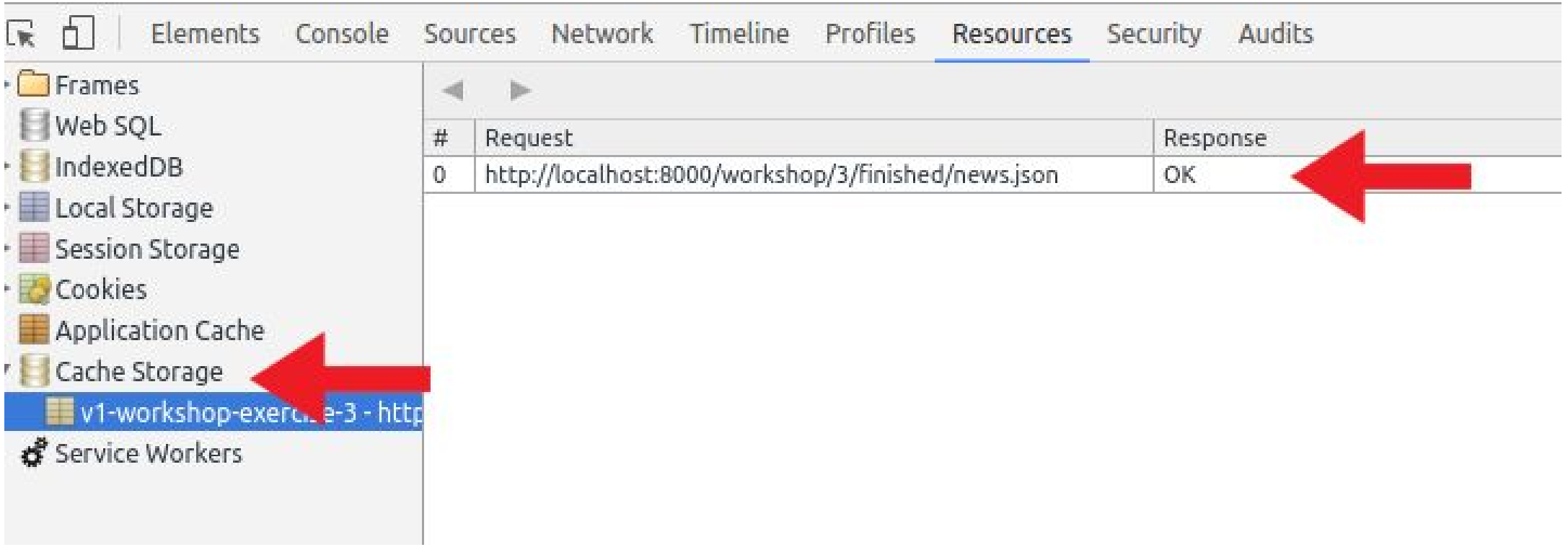
Section recap

- Intercepted a request
- Responded with a hardcoded news object
- Saw hard refresh bypassing the service worker.

3: CacheStorage/Cache API

CacheStorage/Cache API

tooling


Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
Application Cache
Cache Storage
Service Workers
#
Request
Response
0
http://localhost:8000/workshop/3/finished/news.json
OK

The exercise

- This will build on the previous example.
- Instead of hardcoding a response we will fetch the original request.
- We will cache the response to that request.
- We will observe the caching in the response tab of the Chrome web dev tools.
- If you delete your a CacheStorage, delete your ServiceWorker as well.

Some code to look at

```
self.addEventListener('fetch', function (event) {
  if (!event.request.url.endsWith('news.json')) {
    return;
  }
  let fetchedNews =

        fetch(event.request.clone())
          .then((response) => {
            // cache key value pair: {event.request, response}

            // clone the response. we will use it again for cache
            return response.clone();
          });

  // fetchedNews is a Promise that needs to resolve to a Response
  event.respondWith(fetchedNews);
});
```

Some code to look at

```
let fetchedNews =

      fetch(event.request.clone())
        .then((response) => {
          // cache a key value pair: {event.request, response}

          // clone the response. we will use it again for cache
          return response.clone();
      });

// fetchedNews is a Promise that needs to resolve to a Response
event.respondWith(fetchedNews);
```

Writing to the CacheStorage

```
// cache a key value pair: {event.request, response}
caches.open('your-cache-name')
   .then((cache) => cache.put(request, response));
```

Section recap

- Request/Response objects.
- Why you need to clone Request/Response.
- Fetch API
- Caching responses.

4: Return cached request/response

The exercise

- open **workshop/4/start**
- Now instead of just caching the news.json, we will
  - Check if a cached version exists.
  - If it does... return the cached response.
  - If not, fetch the remote version, cache it and return the response.

Some code to look at

```
// to check if a cache exists
caches.match(request)
  .then((response) => {
    if (response) {
      return response;
    }

    return fetch(request.clone())
      .then((response) => {
        cacheResponse(request, response));

        return response.clone();
      });
  });
```

Some code to look at

```
// to check if a cache exists
caches.match(request)
  .then((response) => {
    if (response) {
      return response;
    }
```

Section recap

- Intercept news request
- Check if cache exists
- Return cached data

5: Caching page assets on install

The exercise

- We need an array of assets, a whitelist.
  (The network tab or server log can tell you what you are loading.)
- We want to cache whitelisted page assets during the **‘install’** part of the service worker lifecycle.
- We want to wait until this is done before we allow installation to complete.
- We should be able to see the assets in the CacheStorage.

Some code to look at

```
let whitelistURLs = [ … ]; // An array of strings

self.addEventListener('install', function (event) {
  let whitelistCached =
        caches.open('your-cache-name')
          .then((cache) => cache.addAll(whitelistURLs));

  event.waitUntil(whitelistCached);
});
```

The exercise

- We will still have a hardcoded check for the news.json.
- We need a check for the URI in the whitelist to see if we should try to fetch from the cache or just use default browser behaviour.

Some code to look at

```
self.addEventListener('fetch', function (event) {

  if (!event.request.url.endsWith('news.json') &&
    !inWhitelist(event.request.url)
  ) {
    return;
  }

  ...

  let foundMatch = fromCacheOrNetwork(event.request);

  event.respondWith(foundMatch);
});
```


LET ME GET THIS STRAIGHT,
I CAN ACCESS A WEB APP WHEN NOT ON THE WEB?
memegenerator.net

Section recap

- Cache a whitelist of assets when installing a service worker
- See the cached assets in the CacheStorage tools
- Load whitelisted cached page elements when offline
- Load the news data when offline
- Force load from the server if hard refresh

Challenges

- Security. Storing a user's data unencrypted on a public browser? Shan't.
- Space: How much can we store before the browser purges the application?
  - Clean up after yourself, or your whole domain could be purged.
- Browser adoption for service workers.
  - Fallback to appcache the predecessor that is very easy to get wrong.
- Indexeddb has much better browser support.

Useful reading materials

http://offlinefirst.org
https://github.com/slightlyoff/ServiceWorker/blob/master/explainer.md
http://www.html5rocks.com/en/features/offline
https://jakearchibald.github.io/isserviceworkerready/
https://jakearchibald.github.io/isserviceworkerready/resources.html
https://developers.google.com/web/updates/2015/12/background-sync?hl=en
http://codepen.io/justgooddesign/pen/ygiaJ
http://www.html5rocks.com/en/tutorials/service-worker/introduction/
https://github.com/mozilla/localForage
https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers
https://blog.wanderview.com/blog/2015/10/13/patching-resources-in-service-workers/
https://developer.mozilla.org/en/docs/Online_and_offline_events
https://jakearchibald.com/2014/offline-cookbook/
https://serviceworke.rs/index.html ← great cookbook by mozilla
https://hacks.mozilla.org/2015/11/offline-service-workers/
https://youtu.be/1FWUYHxt5W4 ← firefox's new ServiceWorker debugging tools
https://github.com/GoogleChrome/sw-toolbox
https://www.talater.com/upup/
https://mdn.mozillademos.org/files/12638/sw101.png ← Cheat sheet
http://blog.vanamco.com/indexeddb-fundamentals-plus-a-indexeddb-example-tutorial/
https://brandonrozek.com/2015/11/limiting-cache-service-workers-revisited3/

https://notifications.spec.whatwg.org/
http://hood.ie/


its learning
AT THE HEART OF EDUCATION