**Lesson 3 Start a Web Service Worker**

* Lesson 3.1 Installing service worker
* Lesson 3.5 Service worker, lifcycles
* Lesson 3.10 Intercepting the request
* Lesson 3.15 Listening for the response, for the fetch
* Lesson 3.17 - an install chache event
* Lesson 3.18 - Serving the files from the cache
* Lesson 3.20 Active service worker, active cache events, Scaling cache changing
* Lesson 3.23 Service Worker Lifecycle registering events
* Lesson 3.25 indexController.js Communicates with the sw.j

**1. C:\Users\Lenovo\wittr\public\js\sw\index.js** is where we can write code, what a service register should do.

**2.** **/*Installing service worker, + register a simple service*\*/**

**public/js/main/index controller.js**

* Responsible for the inital code, registering the web worker
* Best practice => is to name the submethods, of an object as .\_methodName if it is onlycalled once by it's object.

**Register a sevice worker at the index controller (has a promise, which can be catched)**

navigator.serviceWorker.register('/sw.js').then(function(reg) {

console.log('Worker has been registered');

if (!navigator.serviceWorker.controller) { // So it will check if the browser enables the usage of a service worker.

return;

}

}).catch(()=>{

console.log('Error has been occured');

});

**Add a simple service**

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

console.log('yo',event.request);

});

**3. *Service worker, lifcycles/(*Lesson 3.5 *)***

* When the html page loaded firstly and the serviceworker.js secondly, all of the request will be avoiding the serviceworker, for the first time, until it has been set up but when the page is refreshed, all of the requests will go through it.
* If serviceworker has changed, than it is waiting for a completely new url refresh, to take over the new package (old and new service workers are not overlapping eachother) (Force update on page load)

**Bypass service worker (Shift + Refresh )**

**Lesson 3.7**

By adding a breakpoint in the devtools, you can inspect all of the states of the ojbect in the js file .

**LESSON 3.10 Intercepting the request (sw.js) ///**

**event.respondWith**, we can totally controll the stuff what is sent back to the page

**response (string, object)**

With the object you can modify the headers as well .

So we can intercept with proper html if we insert

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

event.respondWith(

new Response('<strong class="a-winner-is-me" >Hello from your friendly neighbourhood service worker!</strong>', {

headers: { 'Content-Type': 'text/html' }

})

);

});

**//Lesson 3.11-12 using fetch to respond for queries + check the url string (sw) //**

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

var id = event.request.url;

var isJpg = id.substr(id.length - 4);

if(isJpg == '.jpg'){ //Quicker way to do it --> event.request.url.endsWith('.jpg')

event.respondWith(

fetch('/imgs/dr-evil.gif')

);

}

});

**Lesson 3.15 Listening for the response, for the fetch (sw) //**

- We can fetch complete event requests.

- Fetch has a promise, which is a response, we can check it's properties

- Catch is a callback after fetch,then.catch if sth went wrong (offline databases)

- U can return another fetch as well.

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

event.respondWith(fetch(event.request).then(function(response){

//You can check the status of the response

if (response.status == 404){

return fetch('/imgs/dr-evil.gif');

}

return response;

}).catch(function(){

return new Response('Offline stuff!');

});

);

});

**Lesson 3.17 - an install chache event (sw) //**

* cache.open ('cacheName').then().catch(); --> Will open the cache + return a promise with the cache
* cache.put(request,response); ---> will add a singe cache request
* cache.addAll([]); ---> Atomic - has to match all!
* cache.match(request); --->Will find the match in the given cache return a promise or null
* caches.match(request); ---> will look up in all of the caches

**Add files to the cache.**

self.addEventListener('install', function(event) {

event.waitUntil(

caches.open('wittr-stat-cache1').then(function(cache) {

return cache.addAll([

'/skeleton',

'js/main.js',

'css/main.css',

'imgs/icon.png',

'https://fonts.gstatic.com/s/roboto/v15/2UX7WLTfW3W8TclTUvlFyQ.woff',

'https://fonts.gstatic.com/s/roboto/v15/d-6IYplOFocCacKzxwXSOD8E0i7KZn-EPnyo3HZu7kw.woff'

]);

})

);

});

**Lesson 3.18 - Serving the files from the cache (sw.js) //**

If the request event can be found in the cache, it should serve it from there first.

event.respondWith(

caches.match(event.request).then(function(response){

if(response) return response;

return fetch(event.request);

})

);

});

**/Lesson 3.20 Active service worker, active cache events, Scaling cache changing (sw.js + main.css)//**

**"Active" state of a service worker**

self.addEventListener('activate', function(event) {});

**Activate cache event:**

* caches.delete(cacheName) --> Will return promises when it is done.
* caches.keys() --> Makes us enable to loop over the cache items

**Scaling the cache changing event**

Create a variable for the current static cache and use it the cache events

var staticCacheName = 'wittr-static-v7';

var contentImgsCache = 'wittr-content-imgs';

var allCaches = [

staticCacheName,

contentImgsCache

];

self.addEventListener('install', function(event) {

event.waitUntil(

caches.open(staticCacheName).then(function(cache) {

return cache.addAll([

'/skeleton',

'js/main.js',

]);

})

);

});

Ten looping over all of the caches and delete everyone which is uneccessary

self.addEventListener('activate', function(event) {

// - Getting all of the keys, which are promises, with Promise all we wait until the loop will be finished for every promise

// - Then we filterthe cachenames, when it is done and an array is returned

// - We loop all of the element's in the array and will delete them

event.waitUntil(

caches.keys().then(function(cacheNames){

return Promise.all(

cacheNames.filter(function(cacheName){

return cacheName.startsWith('wittr-') && cacheName !=staticCacheName;

}).map(function(cacheName){

return caches.delete(cacheName);

})

);

})

);

});

**Lesson 3.23 Service Worker Lifecycle registering events (indexController.js)**

* reg.unregister() --> will unregister the current sw
* reg.update() --> will update the current sw
* reg.installing() --> will installing it (it has a state, installing/installed)
* reg.waiting() --> SW is waiting to replace the current one
* reg.active() --> Sw is active
* reg.addEventListener('updatefound',function(){ }); --> add a listener, to continously looking for changes
* reg.installing.addEventListener('statechange', function(){ })--> will listen to the install state changes

navigator.serviceWorker.register('/sw.js').then(function(reg) {

// Check service worker browser support

if(!navigator.serviceWorker.controller){

}

// TODO: if there's an updated worker already waiting, call

if(reg.waiting){

indexController.\_updateReady();

}

// Listen for the installing, once it has been finished, run a code

if(reg.installing){

reg.installing.addEventListener('statechange', function(){

if(this.state =="installed"){

indexController.\_updateReady();

}

});

indexController.\_updateReady();

}

// Listen for any update

reg.addEventListener('updatefound', function(){

reg.installing.addEventListener("statechange",function(){

if(this.state =="installed"){

indexController.\_updateReady();

}

});

});

});

};

**Lesson 3.25 indexController.js Communicates with the sw.j**

\* listening to any controllerchange

navigator.serviceWorker.addEventListener('controllerchange',function(){

});

\*Page reload :

window.locate.reload();

\*Sending an message from the indexController.js registering

reg.installing.postMessage({foo:'bar'});

\*Listening in the sw.js file for the changes

self.addEventListener('message', function(event){

event.data

});

\* Telling to a current sw to skip waiting

self.skipWaiting();

\*Example of Sending & Listening to a message

//IndexController sending

worker.postMessage({action: 'skipWaiting'})

//sw.js listening

self.addEventListener('message', function(event) {

if (event.data.action === 'skipWaiting') {

self.skipWaiting();

}

});