داکیومنت اموزش PWA

* 1. درس 7 Manifest.json

مانیفست به همراه SW کار تنظیمات ابتدای برنامه ما را انجام میدهد. در واقع در زمان ابتدای شروع برنامه SW تنظیمات اولیه از مانیفست می گیرد.



* 1. در 8 Manifest.json



* 1. درس 9 نصب شبیه ساز گوشی همراه
  2. درس 10 تنظیمات IOS



* 1. درس11 تنظیمات برای IE وEdge



* 1. درس 12 معرفی SW.js

در این جلسه کار با callback.js را کار کردیم. اینکه چطور به return داخل یک تابع دسترسی داشته باشیم. در شکل زیر ما به return داخل تابع دسترسی نداریم. یعنی خط consol اجرا نمیشه:



برای حل این مشکل از callback به صورت زیر استفاده می کنیم:



* 1. درس 13 مدیریت خطا ها در callback

برای کنترل خطا در callback از دو ورودی به ان استفاده کردیم. اگر بزرگتر از 10 بود خطا برگردد و در غیر این صورت ادامه دهد و در زمان فراخوانی با کنترل آرگومان اول این خطا را کنترل کریم:



* 1. درس 14 آشنایی با Promise

در این درس با Promise اشنا می شویم. در promise ابتدا یک کانستراکتور مشابه نام فایل js می سازیم که با دو ورودی کار می کند. شرط داخل تابع هم که مشخص ست. زمان فراخوانی تابع add اگر شرط درست باشده then و اگر اشتباه باشد catch اجرا می شود. با این روش از جهنم callback راحت می شویم و کنترل خطا رو راحتر انجام میدهیم.



* 1. درس 15 Promise.all

در مورد promise اول اینکه به صورت زیر می توان مدیریت خطا کرد و از شر جهنم callback راحت شد.



مورد بعدی هم promise all هست که در این مورد می توان یک تابع نوشت که دو ورودی داشته باشد به صورت ارایه بگیرد. دو تابع add,minus به عنوان ورودی به promiseall داده شدند. حالا then ,catch خروجی دو تابع را نشان می دهند(البته اگر هر دو resolve باشند. اگر یکی از توابع reject باشه، کلا catch نمایش داده میشه.). این مورد زمانی که بخواهیم خروجی 2 تا api را مقایسه کنیم به درد می خورد.



* 1. درس 16 بررسی Promis Race

در این دستور هر کدام از دوتابع که زودتر خروجی بدهند، خروجی ان نمایش داده می شود. در این شکل Add چون سریعتر انجام می شود خروجی ان نمایش داده می شود:



* 1. درس 17 XML Http Request

معرفی این سایت برای دریافت داده از api ها: <https://reqres.in/>

در فایل fetch.js به صورت زیر api را فراخوانی میکنیم. ابتدا یک کانکشن میسازیم. بعد کانکشن را باز می کنیم. نوع پاسخرا مشخص می کنیم. Onload برای این است که زمانی که کار گرفتن دیتا تمام شد. Onerroe برای زمان خطا و send دستور را اجرا می کند.



* 1. درس 18 معرفی و استفاده از Fetch

در این قسمت روش استفاده از fetch بیان شد. Fetch نوعی promise هست پس then,catch دارد. دلیل then صورتی رنگ اینه که پاسخ response.json خودش یک promise هست.



* 1. درس 19 نمایش اطلاعات API در PWA

در این قسمت اطلاعات رو از API گرفتیم و نمایش دادیم:





* 1. درس 20 لود کردن فایل با Fetch

در این قسمت سعی کردیم با fetch یک فایل html را بگیرم و نمایش بدهیم. این کار می توان برای نمایش یک المان مثل نوتیفیکیشن و متن و ... که می خواهیم در تمامی صفحات تکرار شود، استفاده کرد.







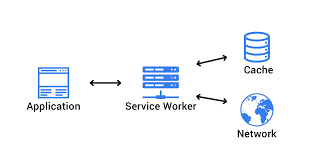
* 1. درس 21 Post و Put با Fetch

در این جلسه بقیه موارد API مثل Post رو بررسی کردیم:



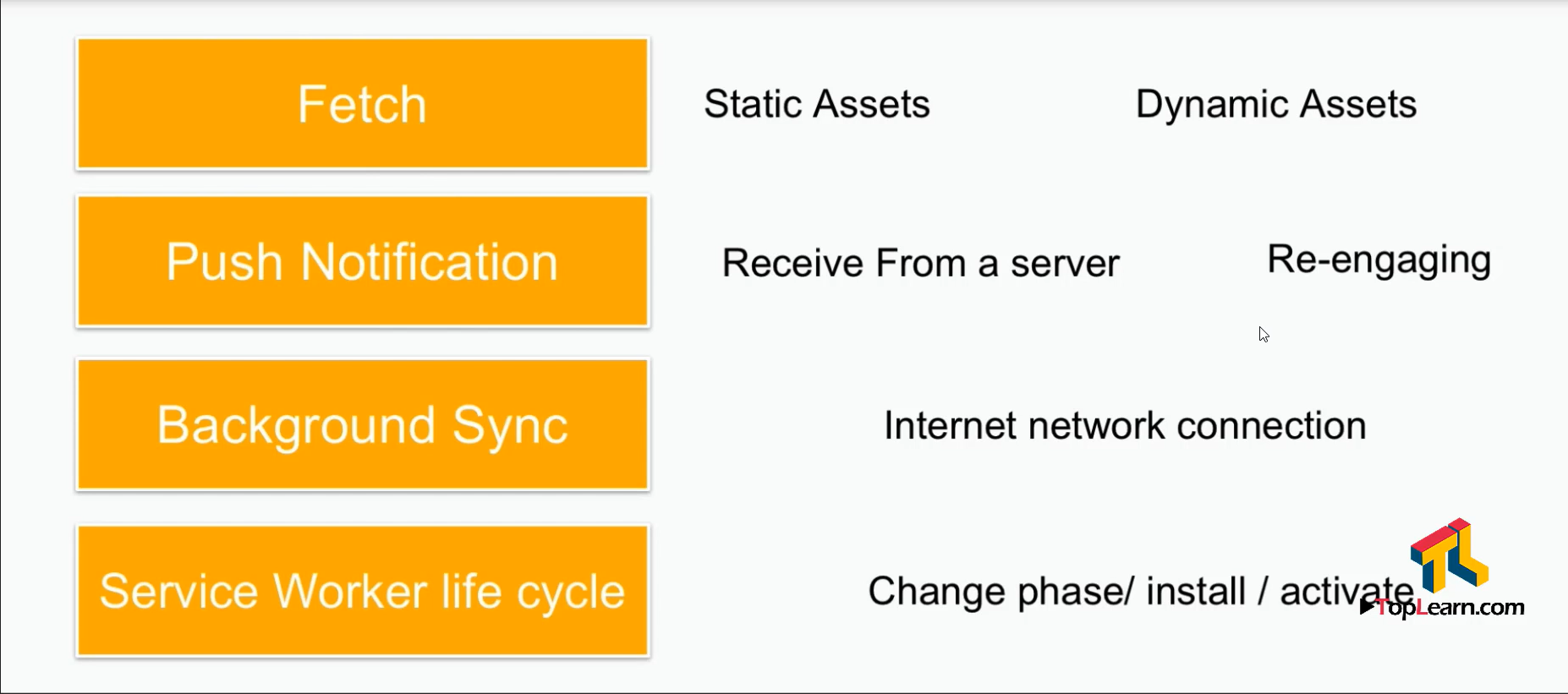
* 1. درس 22 Service Worker چیست؟

این مفهوم رابط بین application و وب و catch هست. این صفحه Script های صفحات را کنترل و ران می کند و مثل عکس به عنوان واسطه عمل می کند.



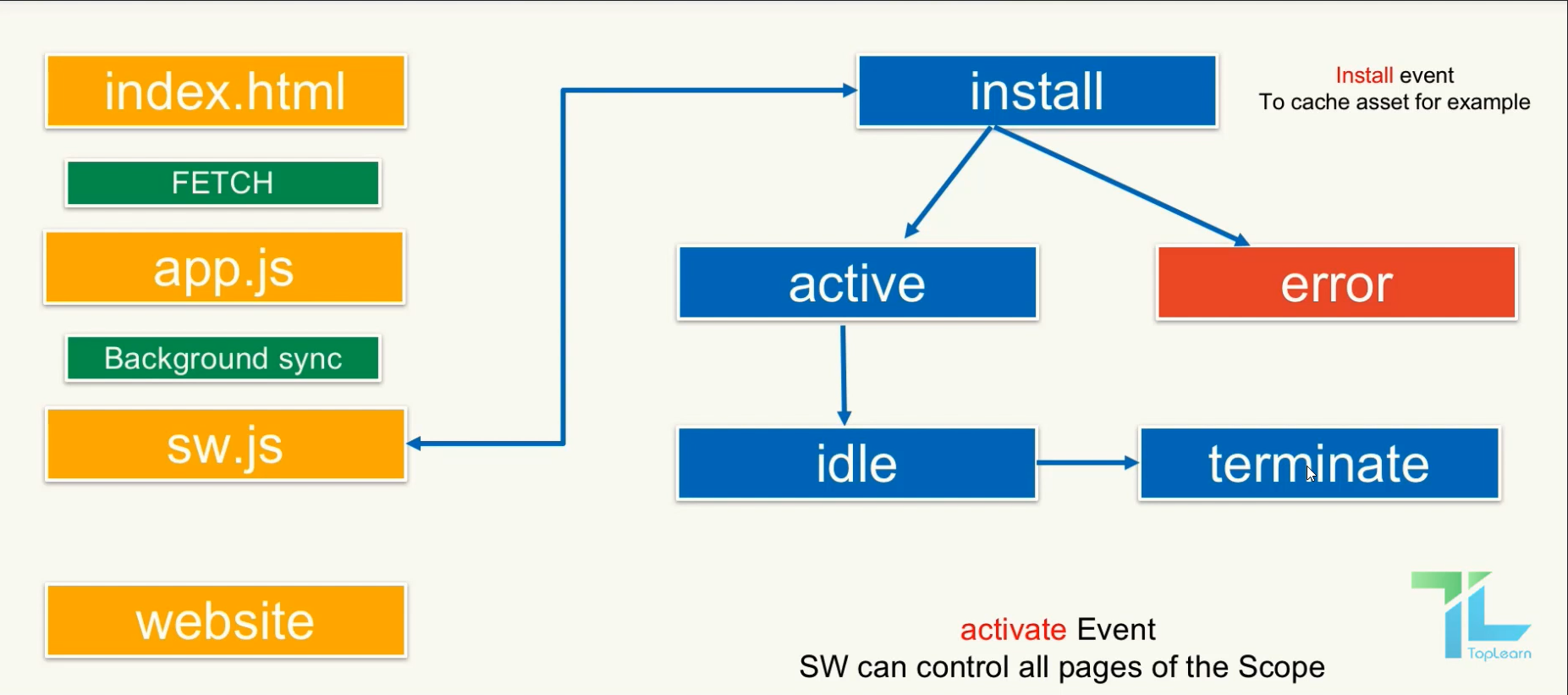
* 1. درس 23 ویداد های Servise Worker

در این جلسه رویداد های سرویس وروکر رو به صورت تئوری بررسی کردیم.



* 1. درس 24 چرخه حیات Servise Worker

در این جلسه چرخه کار سرویس ورکر به صورت تئوری بررسی شد.



* 1. درس 25 ساخت و install کردن Service Worker

در این قسمت اسکوپ بررسی شد. اسکوپ به سرویس ورکر می گوید که کدام قسمت هارا باید کنترل کند.

Sw.js رو همیشه توی روت پروژه بزارین که همه فایل و صفحات را کنترل کند. فایل Html اصلی پروژه که به صورت زیر هست.



فایل app.js که توی اون سرویس وروکر رو ریجیستر می کنیم. در شرط if صورتی چک می کنیم اگر مروگر از سرویس ورکر پشتیبانی می کنه بعد ریجیستر بشه. قسمت ابی ه برای این است که ابتدا تمام محتوای صفحه لوود بشه بعد سرویس روکر ریجیستر بشه و اگر مشکلی در لوود بود سرویس روکر ریجیستر نشه. بقیه موارد هم که مثل promise هستند:



خود سرویس ورکر هم به صورت زیر تعریف میشه و به این صورته Event های مختلف داخل اون نوشته میشه مثل install,activate که در جلسه پیش بررسی کردیم:



* 1. درس 26 مدیریت service Worker از طریق مروگر

در این قسمت درستور fetch به SW اضافه شد:

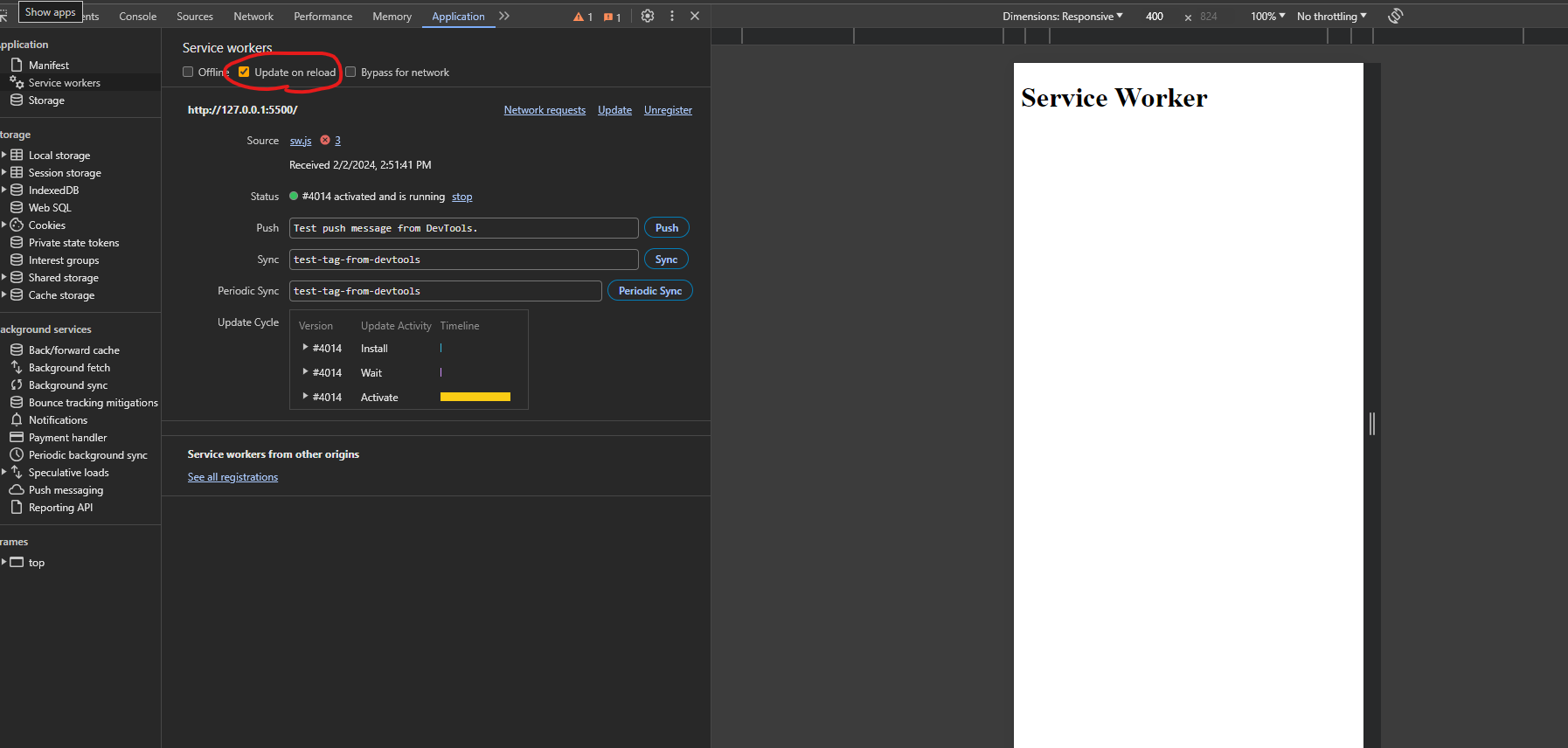


* 1. درس 27 دیباگ کردن service Worker

در این قسمت با استفاده از کد هایلایت شده مرورگر رو مجبور می کنمیم که زمانی که SW تغییر کرد به صورت خودکار خودش sw جدید را ریجیستر بکنه. البته بهتره بعد از چند رو این کد رو برداریم. یکسری مشکل ایجاد میکنه.



در زمان Develop هم این کار میشه کرد:



* 1. درس 28 بررسی App Banner

در این قسمت app Banner یا manifest بررسی شد و هم چنین ارتباط بین کروم با کروم شبیه ساز بررسی شد.

* 1. درس 29 نصب Peompt App Banner

در این قسمت اجازه نصب برنامه رو به صورت دستی گرفتیم و روی یک دکمه گزارشتیم. این کار با prompt انجام شد.

برای بررسی جلسه بهتره که جلسه رو دوباره ببینی.



* 1. درس 30 دیباگ PWA در Emolator

دراین قسمت طریقه اتصال امولاتور گوشی رو به کروم دکستاپ بررسی کردیم. جلسه مهمی هست برای دیباگ روی گوشی بسیار کار امد هست.

* 1. درس 31 ساخت SW پروژه نهایی

در این قسمت در پروژه اصلی SW ریجیستر می شود و نوشته می شود:

قسمت صورتی منتظر می ماند که کل صفحه لود شود. Scope دامنه کنترل SW را مشخص می کند که اینجا کل Root است. در قسمت log می توان res را مشاهد کرد.

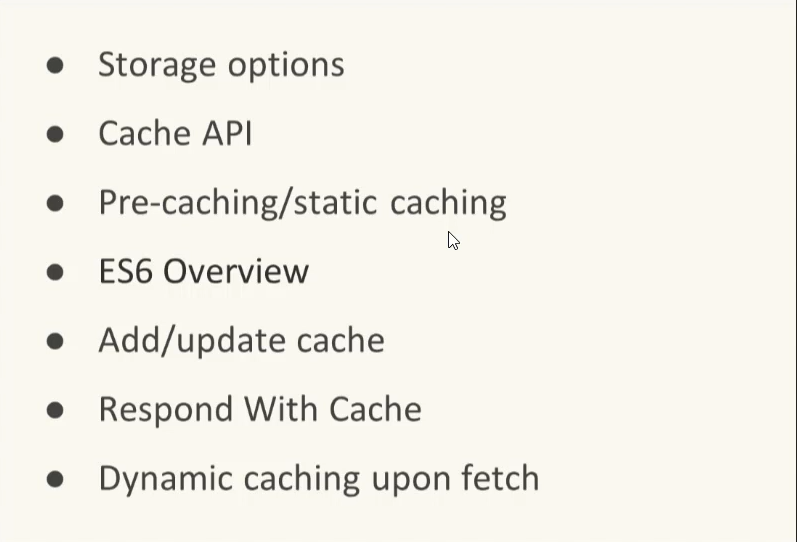


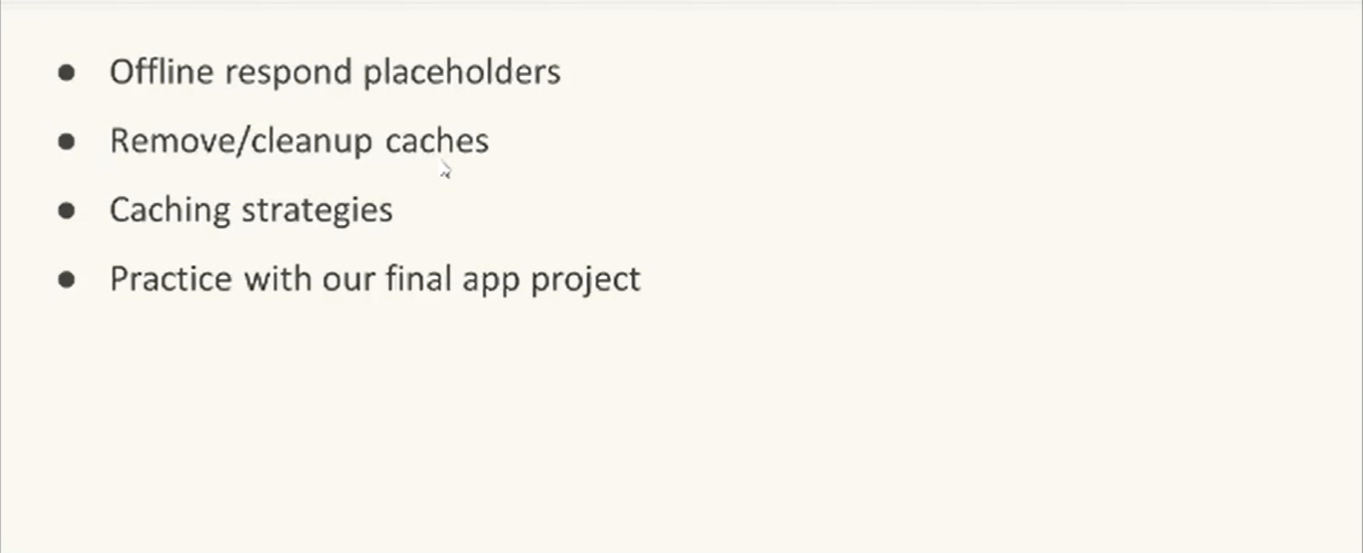
SW.js را ه به صورت زیر می نوسیم:



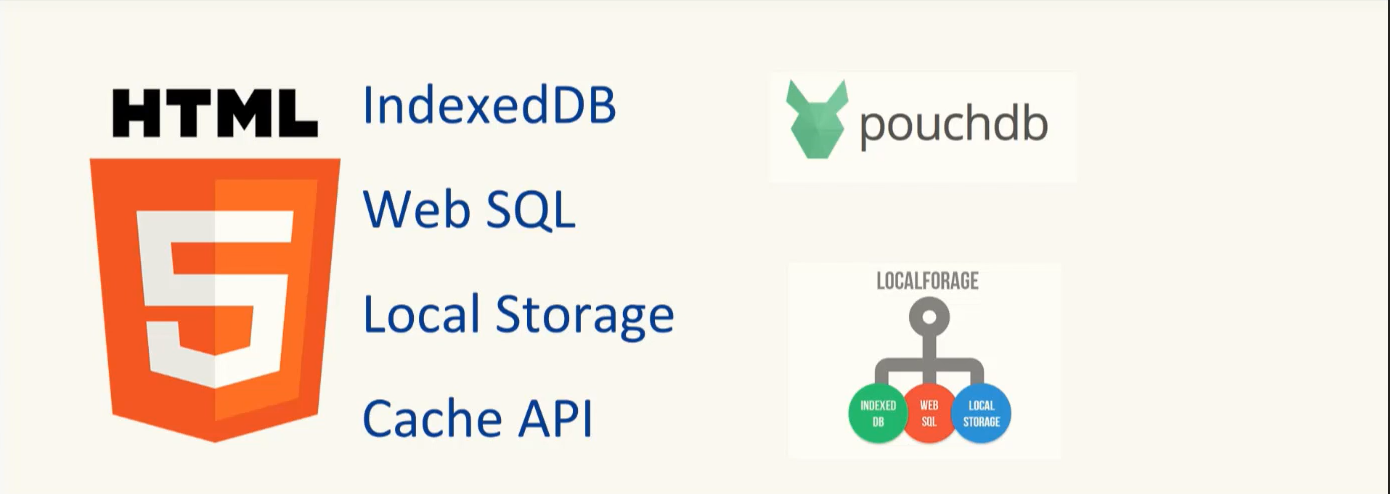
* 1. درس 32 ذخیره سازی اطلاعات (Storage Options)

در این جلسه توضیحاتی در مورد کش داده شد. که در زیر لیست شده اند:





در شکل زیر هم دیتابیس های مروگر که پیشنهاد میشه برای PWA استفاده بشه اومده.



* 1. درس 33 بررسی cache Api و PreChachung

در این مبحث cache و cache api بررسی شد و توضیحاتی داده شد.

* 1. درس 34 قابلیت های جدید EC6

در این درس مواردی مثل var,let,const, Arrow function در جاوااسکریپ توضیح داده شد.



* 1. درس 35 cache کردن assets

در این جلسه در مورد cache صحبت شد و اینک چطور صفحات خودمون رو کش کنیم تا در زمان offline استفاده کنیم:



برای باز کردن کش و ذخیره صفحات همان طور که در قسمت ابی گفته شده یک اسم برای کش تعیین می کنیم که با تغییر این اسم کش تغییر می کنه و دوباره از سرور گرفته میشه. قسمت ارایه Assets هم برای ذخیره چندین صفحه است.

قسمت قرمز هم برای این است که از انجایی که جاوااسکریپ غیر هم زمان است این تابع زمان میدهد که کش انجام شود بعد به ادامه کار بپردازد.

با دستور cache.open هم کش را باز می کنیم و با متد addall صفحات را اضافه می کنیم.

* 1. درس 36 Respond cache در PWA

در این درس نمایش کش را بررسی می کنیم. ابتدا درخواس را میگیرم (صورتی). قسمت ابی مشخص می کند در حالت پاسخ و جواب باشد و درحالت درخواست نباشد. قسمت سبز هم کش را نشان میدهد. کش را انجام بده.



* 1. درس 37 Dynamic cache در PWA

در این قسمت Dynamic cache کار شد

const STATIC\_CACHE\_VERSION = 'static\_2'

const DYNAMIC\_CACHE\_VERSION = 'dynamic\_2'

const STATIC\_ASSESTS = [

    '/',

    '/index.html',

    '/help.html',

    '/css/skeleton.css',

    '/css/style.css',

    '/js/main.js'

];

self.addEventListener('install', (event) => {

    console.log('[SW] installing Service Worker ...');

    event.waitUntil(

        caches.open(STATIC\_CACHE\_VERSION)

            .then((cache) => {

                console.log('cache ready');

                return cache.addAll(STATIC\_ASSESTS);

            })

            .catch(e => {

                console.log('cache error');

            })

    );

});

self.addEventListener('activate', (event) => {

    console.log('[SW] activating Service Worker ...');

});

self.addEventListener('fetch', (event) => {

    console.log('[SW] fetching ...');

    const request = event.request;

    event.respondWith(

        caches.match(request)

            .then((response) => {

                return response || fetch(request)

                    .then((res) => {

                        caches.open(DYNAMIC\_CACHE\_VERSION)

                            .then((cache) => {

                                cache.put(request, res);

                            });

                        return res.clone();

                    })

                    .catch((err) => {

                        console.log('[SW] cache fetch error');

                    });

            })

            .catch(console.error)

    );

});

* 1. درس 38 مدیریت خطا در زمان افلاین بودن PWA

در این قسمت مدیریت خطا ها درکش بررسی شد که اگر dynamic یا static مشکل داشت چطور خطا رو کنترل کنیم:



* 1. درس 39 پاکسازی cache

در این جلسه پاکسازی کش های قدیمی رو انجام میدم. به صورتی که اگر کشی از قدیم مانده بود پاک می کنیم:



* 1. درس 40 استراتژی های مهم cache در PWA

در این درس استراتژی های مهم کش بررسی شد. که 4 نوع هستند.:



* 1. درس 41 استراتژی های مهم cache در PWA بخش دوم

در این درس ادامه این استراتژی ها بررسی شد.



* 1. درس 42 پیاده سازی cache در پروژه اصلی

در این جلسه مواردی که برای پیاده سازی روی کش اموزش دیده بودیم را روی پروژه اصلی پیاده کردیم.

دقت شود هیچ موقه SW.js را کش نکنید.

* 1. درس 43 پیاده سازی cache در پروژه اصلی قسمت دوم

در این قسمت پیاده سازی کش کامل شد.

برای فونت های گوگل هم یک داینامیک کش گذاشته شد.

* 1. درس 44 اشنایی ونصب Indexed DB در PWA

در این جلسه این دیتابیس نصب شد.



* 1. درس 45 Insert کردن دیتا در IndexedDB

نوشتن اطلاعات روی دیتابیس(صورتی).

کنترل کردن اینکه مروگر از این دیتابیس پشتیبانی می کند(آبی).

گرفتن اطلاعات از api (سبز).



* 1. درس 46 خواندن اطلاعات از IndexedDB

در این جلسه خواند اطلاعات از دیتابیس بحث شد. خواند با ایدی و خواندن کل داده ها.



* 1. حذف اطلاعات از IndexedDB

در این جلسه حذف بررسی شد:



* 1. درس 48 پیاده سازی IndexedDB روی پروژه نهایی

در این جلسه روی پروژه نهایی دیتابیس نصب شد و اضافه کردن یک note به دیتابیس پیاده شد.



* 1. درس 49 پیاده سازی IndexedDB روی پروژه نهایی PWA

در این درس Delete و Edit به پروژه اصلی اضافه شد.

* 1. درس 50 معرفی اماده سازی FierBase