

آموزش پیاده سازی وب اپلیکیشن پیش رونده PWA

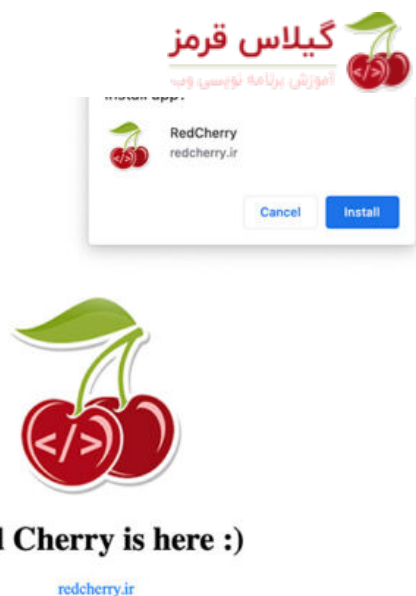


توسط علی رحیمی در تاریخ 15 خرداد 1400

وب اپلیکیشن پیش رونده (Progressive Web App) یا به اختصار PWA این امکان رو به ما میدهد که از وب سایتمون یه نسخه اپلیکیشن ایجاد کنیم. این مبحث وقتی توی ایران پر رنک شد که شرکت اپل برنامه های مالی ایران رو محدود کرد و دیگه نمیشد اپلیکیشن های ایرانی ios رو روی گوشی های آیفون نصب کرد. چاره کار همین PWA شد.

وب سایت ها خدمات تحت وب خودشون رو با استفاده از PWA تونستن روی گوشی های آیفون ارائه بدن. ولی PWA فقط برای گوشی های آیفون نیست. در اصل وقتی PWA روی یه وب سایت اینترنتی پیاده میشه، اون وبسایت قابلیت این رو پیدا میکنه که مثل یه اپلیکیشن روی گوشی یا کامپیوتر نصب بشه. با این کار دیگه نیازی نیست هربار کاربر آدرس سایت رو توی مرورگر وارد کنه تا بتونه از خدمات اون سایت استفاده کنه. بلکه کافیه روی آیکون برنامه در دسکتاپش کلیک کنه تا وب سایت شبیه به یه اپلیکیشن باز بشه.

این آموزش روی همه ی وب سایت ها جواب میده و فرقی نمیکنه از چه زبان و فریمورک یا مدیریت محتوایی استفاده میکنید. فقط روی سایتتون https باید داشته باشید و همه ی لینک هاتون باید از http ریدارکت بشه به https. شما میتونید فایل های این آموزش رو از صفحه گیت هابم به آدرس <https://github.com/alirahimi818/simple-PWA> دانلود کنید، شخصی سازی کنید و روی پروژه خودتون پیاده سازی کنید. (این آموزش طبق فایل های **مخزن گیت هاب** هستش) برای دیدن پیش نمایش این پروژه کافیه به آدرس <https://alirahimi818.github.io/simple-PWA> برید.



ویژگی و امکانات PWA در این آموزش

توی این آموزش یاد میگیرید که چطور PWA رو روی هر سایتی پیاده کنید. با مفهوم و کاربرد هر تیکه از کد آشنا میشيد. وب سایتتون با هر درخواست کاربر، بصورت کامل کش میشه. وب اپلیکیشن بصورت آفلاین کار میکنه. موقع آفلاین بودن کاربر، هر جایی که کش وجود نداشت، صفحه آفلاین رو نمایش میدیم.

ساخت و پیاده سازی آیکون برنامه

اول از همه شما نیاز دارید که ۸ سایز مختلف آیکون برای وب اپلیکیشنتون تهیه کنید. این کار برای اینه که وب اپلیکیشن شما روی هر دستگاهی (مثلا آیفون، سامسونگ و...) که نصب شد، سایز آیکون مورد نظرش موجود باشه. البته بعضی از این آیکون ها در فاوآیکون سایتتون استفاده میشه.

- سایز 16×16 پیکسل <= فایل : fav-16.png
- سایز 32×32 پیکسل <= فایل : fav-32.png
- سایز 57×57 پیکسل <= فایل : fav-57.png
- سایز 72×72 پیکسل <= فایل : fav-72.png
- سایز 114×114 پیکسل <= فایل : fav-114.png
- سایز 144×144 پیکسل <= فایل : fav-144.png
- سایز 192×192 پیکسل <= فایل : fav-192.png
- سایز 512×512 پیکسل <= فایل : fav-512.png

مهم نیست این آیکون ها رو کجای سایتتون آپلود میکنید. ولی جاهایی که از این آیکون ها استفاده میشه رو باید با آدرس خودتون جایگزین کنید. یه سری از این آیکون ها رو با استفاده از کد زیر به تگ <head> سایتتون اضافه کنید:

```
<link rel="shortcut icon" href="img/fav-16.png">
<link rel="icon" type="image/png" sizes="32x32" href="img/fav-32.png">
<link rel="icon" type="image/png" sizes="16x16" href="img/fav-16.png">
<link rel="apple-touch-icon" href="img/fav-57.png">
<link rel="apple-touch-icon" sizes="114x114" href="img/fav-114.png">
```

دو مورد تگ هم برای استایل دادن برنامه توی بعضی از مرورگر ها استفاده میشه که میتونید تنظیمشون کنید (کد رنگ سایتتون رو وارد کنید):

```
<meta name="msapplication-TileColor" content="#af0a1a">
<meta name="theme-color" content="#ffffff">
```

منا تگ اول وقتی سایت شما به استارت ویندوز ۸ پین میشه، این رنگ رو میخونه. منا تگ دوم توی بعضی از مرورگر های گوشی، قسمت آدرس بار رو رنگی میکنه. این ۲ متاتگ اختیاریه و میتونید ازشون استفاده نکنید.

اضافه کردن فایل manifest وب اپلیکیشن

این فایل مهمترین بخش برای اضافه کردن PWA به وبسایت هستش. با استفاده از این فایل مرورگر متوجه میشه که سایت شما از PWA پشتیبانی میکنه. این فایل که توی این پروژه به اسم pwa-manifest.json هستش رو داخل فولدر روت (مسیر اصلی) سایتتون قرار بدید. بعد با استفاده از کد زیر، اون رو داخل <head> سایت فراخونی کنید:

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

این فایل داخل خودش تنظیماتی داره که مرورگر ها از اون برای ساخت وب اپلیکیشن PWA استفاده میکنن. کاربرد هر بخش رو براتون توضیح میدم.

پارامترها:

- name** : اسم وب اپلیکیشن رو باید وارد کنید.
- short_name** : اسم کوتاه برنامه رو وارد کنید. (میتونید با مقدار قبلی برابر قرار بدید)
- description** : توضیحاتی کوتاه درباره برنامه وارد کنید.
- icons** : بصورت آرایه دو سائز از آیکون هایی که ساختید رو وارد کنید. (192×192 و 512×512 پیکسل)
- screenshots** : توی این بخش میتونید اسکرین شات هایی از صفحات مختلف برنامه قرار بدید. (اختیاری هستش و من دو مورد رو به عنوان مثال وارد کردم)
- background_color** : رنگ بک گراند مرورگر رو وارد کنید. (این رنگ میتونه شبیه رنگ قالب سایتتون باشه)
- theme_color** : رنگ تم که توی بعضی از مرورگر ها به نوار آدرس داده میشه رو وارد کنید.
- display** : حالت نمایش وب سایت داخل وب اپلیکیشن رو مشخص میکنه که ۴ حالت داره:

- fullscreen : تمام اندازه صفحه به وب سایت اختصاص داده میشه و هیچ چیزی از مرورگر نمایش داده نمیشه.
- standalone : مثل یه برنامه مستقل عمل میکنه. توی این حالت ممکنه سیستم عامل کاربر بعضی از موارد مربوط به ظاهر مرورگر رو حذف کنه ولی معمولاً نوار وضعیت مرورگر رو حذف نمیکنه.
- minimal-ui : شبیه به مورد قبلی هستش. با این تفاوت که دکمه های عقب جلو مرورگر هم ظاهر میشه.
- browser : حالت پیشفرض مرورگر کاربر هستش.

- **orientation** : حالت نمایش سایت (عمودی/افقی) رو میشه تنظیم کرد. (any هر حالتی رو شامل میشه)
- **start_url** : آدرس شروع وب اپلیکیشن رو وارد کنید. یعنی وقتی که برنامه باز شد چی نمایش داده بشه؟ علامت نقطه یعنی مسیر فعلی که توش هستیم درخواست داده بشه. (یه جورایی میشه گفت آخرین جایی که کاربر توی سایت شما بوده)
- **url** : آدرس سایتتون رو وارد کنید.

درد به ادرس همین اسکوپ برمیخیزد. وی اچه بیاری به محدودیت ندارید علامت بعضی . رو برارید.
lang : زبان سایتتون رو وارد کنید. (زبان فارسی fa-IR هستش)
dir : نحوه نگارش سایتتون رو مشخص کنید. (فارسی راست به چپ یا rtl هستش)

سرویس ورکر (Service worker) وب اپلیکیشن

سرویس ورکر یا به اصطلاح کارگر خدماتی، وظیفه پردازش درخواست ها رو داره. طبق استراتژی که ما براش تعریف میکنیم کار میکنه. مثلا وقتی کاربر صفحه ای از سایت رو باز میکنه بهش دستور انجام کش رو میدیم. مرورگر با این کارگر خدماتی ارتباط برقرار میکنه و تبادلات لازم رو با هم انجام میدن.

فایل pwa-sw.js رو به فولدر روت (مسیر اصلی) سایتتون اضافه کنید. ما توی این فایل یه سری متغیر، فانکشن و گوش کننده رویداد (event listener) داریم که براتون توضیح میدم هر کدوم چه کاری انجام میده.

متغیر cache_storage_name

اسم مخزنی که قراره فایل های سایتتون داخل اون کش بشه. هر اسم دلخواهی میتونه باشه.

متغیر start_page

آدرس صفحه ای از سایتتون که، وقتی وب اپلیکیشن باز شد، در همون ابتدا نمایش داده میشه.

متغیر offline_page

آدرس صفحه ای از سایت که وقتی کاربر آفلاین شد، اون صفحه نمایش داده بشه.

متغیر first_cache_urls

آرایه ای از آدرس ها که در همون ابتدای نصب کش میشه.

متغیر never_cache_urls

آرایه ای از آدرس ها که هرگز نباید کش بشن. من توی این آموزش آدرس های private.html و /panel/ و /custom-url/ رو وارد کردم.

گوش کننده install

وقتی کاربر سایت رو باز میکنه (طبق معرفی کارگر به مرورگر که بعدا انجام میدیم)، این فانکشن صدا زده میشه و کش های اولیه رو انجام میده.

گوش کننده activate

بعد از عملیات نصب این فانکشن فعال میشه تا حالت نصب برای کاربر رو فعال کنه. (یکی از کارهاش حذف مخزن قدیمی کش، در صورت وجود هستش)

گوش کننده fetch

هر درخواست از سمت کاربر واکشی (fetch) میشه. این فانکشن استراتژی مارو برای هر درخواست اجرا میکنه. توی این فانکشن از ۲ مدل استراتژی برای وقتی که کاربر آنلاین هستش و وقتی که آفلاین هستش استفاده کردم.

اول از همه آنلاین بودن کاربر چک میشه و اگه آنلاین بود استراتژی First network اتفاق میفته. توی این استراتژی درخواست کاربر اول از سمت سرور درخواست میشه و اگه جواب برگشت، همون جواب به کاربر نمایش داده میشه و کش میشه. ولی اگه کاربر آفلاین بود وارد استراتژی بعدی میشیم. توی این استراتژی میگیریم که درخواست کاربر توی کش جستجو بشه یا سمت سرور هم درخواست بره، هر کدوم که جواب رو برگردوندن به کاربر نمایش داده میشه و مجدد کش میشه. اگه جوابی برگشت داده نشه (یعنی قبلا اون درخواست از طرف کاربر انجام نشده باشه و در نتیجه کش نشده باشه) و کاربر هم برای دریافت از سرور آفلاین باشه، صفحه آفلاین رو به کاربر نمایش میدیم. (توی همه ی این کش کردن ها، استثنای لیست never_cache_urls بررسی میشه)

فانکشن `checkFetchRules`

قوانین قبل از واکنشی رو بررسی میکنه. توی این فانکشن اول بررسی میکنیم که درخواست از دامنه داخلی وب اپلیکیشن باشه (یعنی خارج از سایت درخواست داده نشه). قانون بعدی بررسی میکنه که نوع درخواست حتما http یا https باشه. قانون بعدی نمایش صفحه آفلاین برای درخواست هایی به جز GET هستش (چون نیازی نیست که درخواست های POST کش بشن)

در انتهای این فایل از کارگر خدماتی گوگل به اسم `workbox` برای مدیریت بهتر گوگل آنالتیکس استفاده کردیم. البته کارایی این سرویس ورکر خیلی زیاده که اگه دوست داشته باشید میتونید از **صفحه گوگل** اطلاعات بیشتری به دست بیارید.

فراخوانی سرویس ورکر

خب حالا نیازه که فایل سرویس ورکرمون رو به مرورگر معرفی کنیم. برای این کار من فایلی در مسیر `js/main.js` قرار دادم و با استفاده از کد زیر به `<head>` سایت اضافه کردم. (شما میتونید این فایل رو هر جایی از سایتتون آپلود کنید و آدرسش رو در کد زیر جایگزین کنید)

```
<script type='text/javascript' src="js/main.js"></script>
```

توی این فایل، بعد از لود شدن کامل سایت، در خط چهارم، آدرس فایل سرورس ورکر یا کارگر خدماتیمون (`pwa-sw.js`) رو به مرورگر اعلام میکنیم. یه فانکشن هم برای آنلاین یا آفلاین بودن گذاشتم که وقتی کاربر در لحظه آفلاین شد، با یه پیام بهش میگه که شما آفلاین هستید. (شما میتونید کدهارو طبق امکانات سایتتون تغییر بدید یا کل این فایل رو به فایل جاوا اسکریپت دیگه ای منتقل کنید)

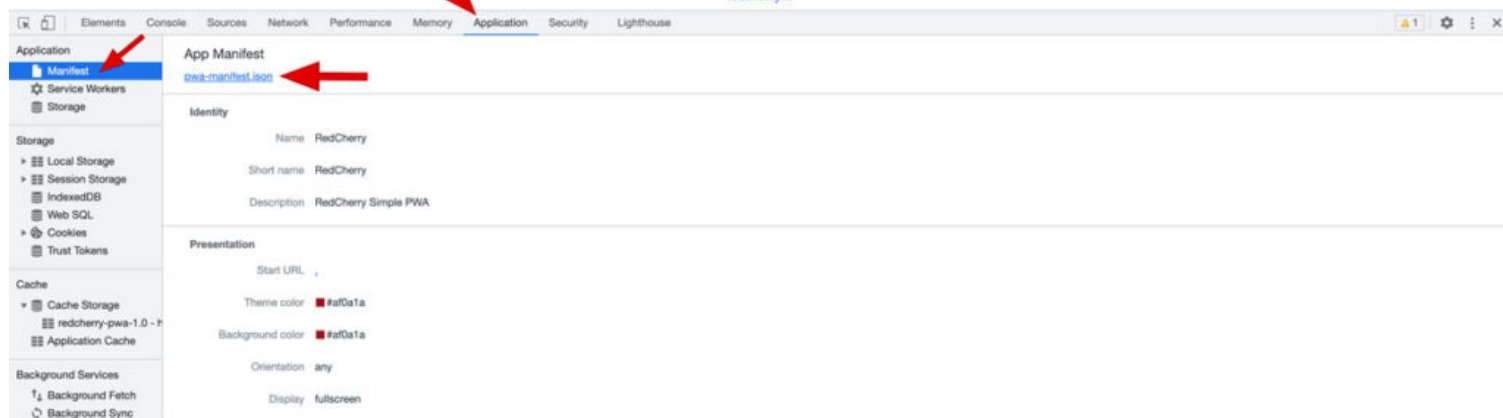
تست کارکرد وب اپلیکیشن

حالا میتونید سایتتون رو داخل گوگل کروم باز کنید. روی صفحه کلیک راست کنید و `Inspect` بگیرید. به تب `Application` برید و روی `Manifest` کلیک کنید. اگه فایل مانیفست رو درست اضافه کرده باشید، اینجا اطلاعاتش رو بهتون نمایش میده. در واقع مرورگر اعلام میکنه که تونسته با مانیفست ارتباط برقرار کنم. اطلاعاتی که داخل فایل `pwa-manifest.json` وارد کرده بودید توی این بخش مشاهده کنید:



Red Cherry is here :)

redcherry.ir

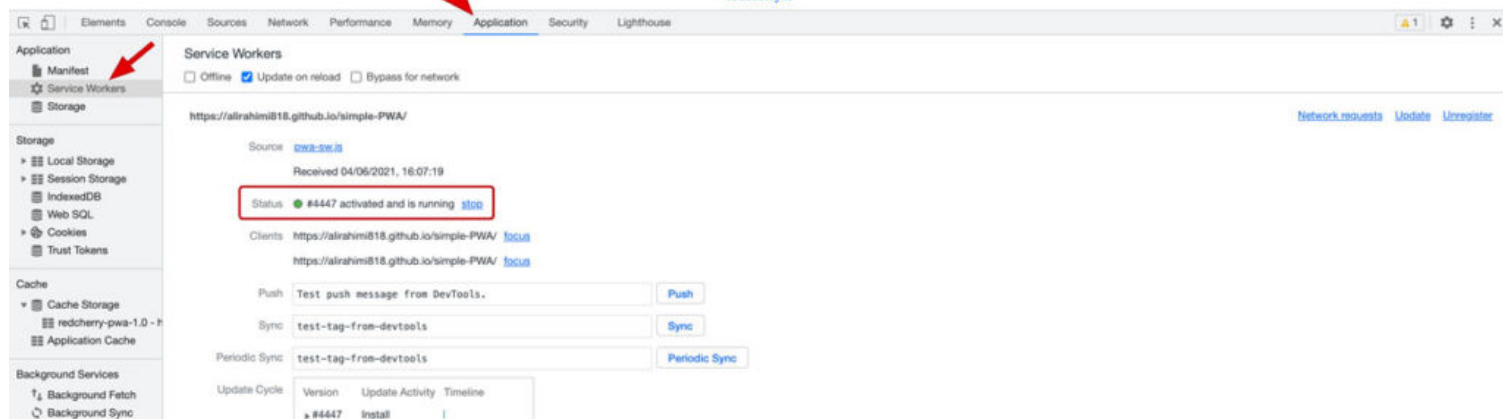


برای بررسی سرویس ورکر از منو سمت چپ روی گزینه Service Workers کلیک کنید. اینجا به شما اطلاعات کارگر خدماتی که به مرورگر معرفی کردید رو نشون میده. وضعیت سرویس همیشه باید Activated and is running (سبز) باشه تا فعالیت کارگر درست انجام بشه.

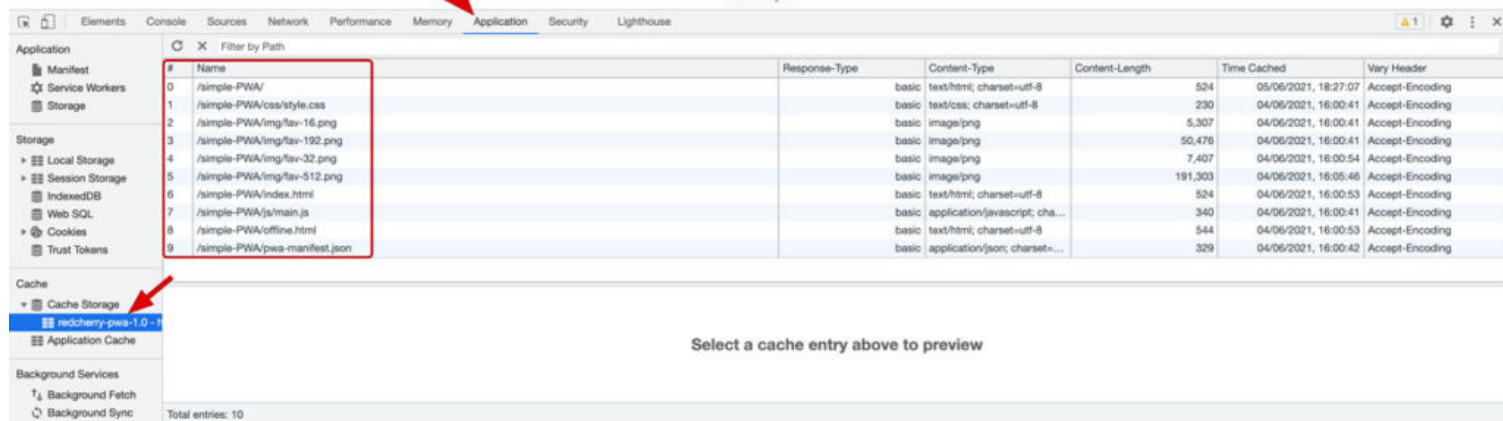


Red Cherry is here :)

redcherry.ir



از منو سمت چپ توی بخش Cache اسم مخزن کشی که معرفی کردید رو باید ببینید. روش که کلیک کنید تمام فایل ها و آدرس هایی که کش شده رو بهتون نمایش میده. دقت کنید که من آدرس private.html رو توی لیست کش نشدن گذاشتم. اگه کاربر توی اون صفحه هم بره ، باز اون صفحه کش نمیشه و همیشه از سمت سرور خوانده میشه. این کار برای سایت های فروشگاه (سبد خرید، لیست علاقه مندی و...) میتونه مفید باشه.

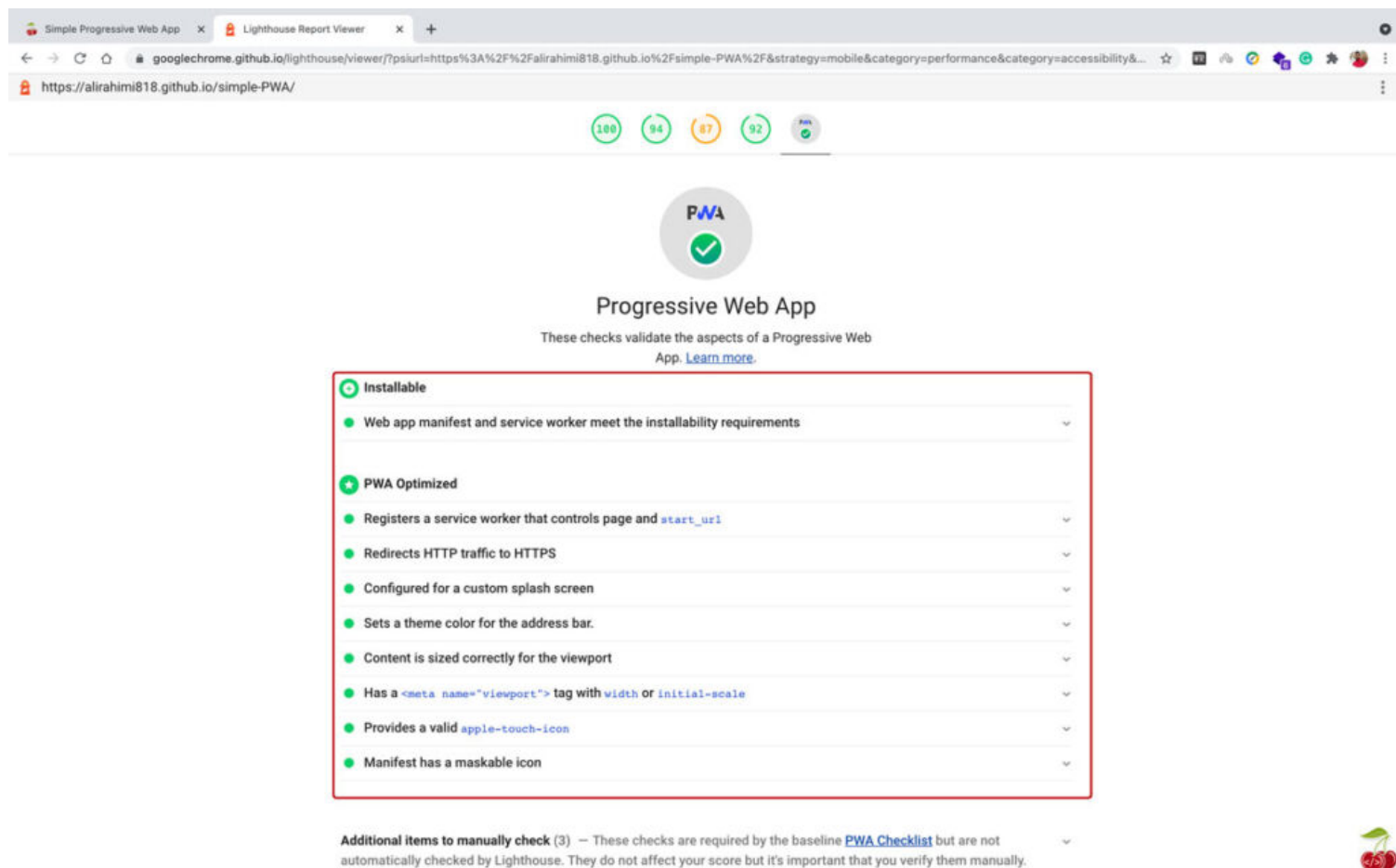


#	Name	Response-Type	Content-Type	Content-Length	Time Cached	Vary Header
0	/simple-PWA/	basic	text/html; charset=utf-8	524	05/06/2021, 18:27:07	Accept-Encoding
1	/simple-PWA/css/style.css	basic	text/css; charset=utf-8	230	04/06/2021, 16:00:41	Accept-Encoding
2	/simple-PWA/img/fav-16.png	basic	image/png	5,307	04/06/2021, 16:00:41	Accept-Encoding
3	/simple-PWA/img/fav-192.png	basic	image/png	50,476	04/06/2021, 16:00:41	Accept-Encoding
4	/simple-PWA/img/fav-32.png	basic	image/png	7,407	04/06/2021, 16:00:54	Accept-Encoding
5	/simple-PWA/img/fav-512.png	basic	image/png	191,303	04/06/2021, 16:05:46	Accept-Encoding
6	/simple-PWA/index.html	basic	text/html; charset=utf-8	524	04/06/2021, 16:00:53	Accept-Encoding
7	/simple-PWA/js/main.js	basic	application/javascript; cha...	340	04/06/2021, 16:00:41	Accept-Encoding
8	/simple-PWA/offline.html	basic	text/html; charset=utf-8	544	04/06/2021, 16:00:53	Accept-Encoding
9	/simple-PWA/pwa-manifest.json	basic	application/json; charset=...	329	04/06/2021, 16:00:42	Accept-Encoding

Select a cache entry above to preview

Total entries: 10

همچنین شما میتونید وب سایتتون رو از طریق **اکستنشن فانوس دریایی (Lighthouse)** چک کنید تا مطمئن شید که تمامی موارد مربوط به PWA رو پیاده سازی کرده باشید.



Simple Progressive Web App x Lighthouse Report Viewer x

googlechrome.github.io/lighthouse/viewer/?psiurl=https%3A%2F%2Falirahimi818.github.io%2Fsimple-PWA%2F&strategy=mobile&category=performance&category=accessibility&...

https://alirahimi818.github.io/simple-PWA/

100 94 87 92

PWA

Progressive Web App

These checks validate the aspects of a Progressive Web App. [Learn more.](#)

Installable

- Web app manifest and service worker meet the installability requirements

PWA Optimized

- Registers a service worker that controls page and `start_url`
- Redirects HTTP traffic to HTTPS
- Configured for a custom splash screen
- Sets a theme color for the address bar.
- Content is sized correctly for the viewport
- Has a `<meta name="viewport">` tag with `width` or `initial-scale`
- Provides a valid `apple-touch-icon`
- Manifest has a maskable icon

Additional items to manually check (3) — These checks are required by the baseline [PWA Checklist](#) but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.

خب کار این آموزش تمومه. اگه سوال و یا مشکلی داشتید، میتونید داخل نظرات همین پست مطرح کنید.

مخزن گیت هاب این آموزش : <https://github.com/alirahimi818/simple-PWA>

