- PWAs

"Using web tech. stds. & **progressive enhancements** to create web apps that look like **native apps**"

- Chrome, Mozilla, opera, samsung, MS

- uses the support of browsers/OSs APIs

- as icon

- even if offline

- fast loading

- fast UX

- external to the browser's UI

- PWAs

Why?

- proprietary development for creating Native apps
- different versions for different platforms for the same app using different languages (Kotlin, Swift ..)
- different store deployment guidelines & wait times
- a native app is faster than a web site
- not limited to the browser UI
- works offline (reliable)
- push support
- linkable, shareable & discoverable
- uses web standards
- UX
- secure (TLS)
- responsive

- PWAs

Evolution

1999: MS HTA

2005: Symbian WRT

2008: Apple iOS home screen web apps

PWA users

Twitter

Flipkart

Uber

Forbes

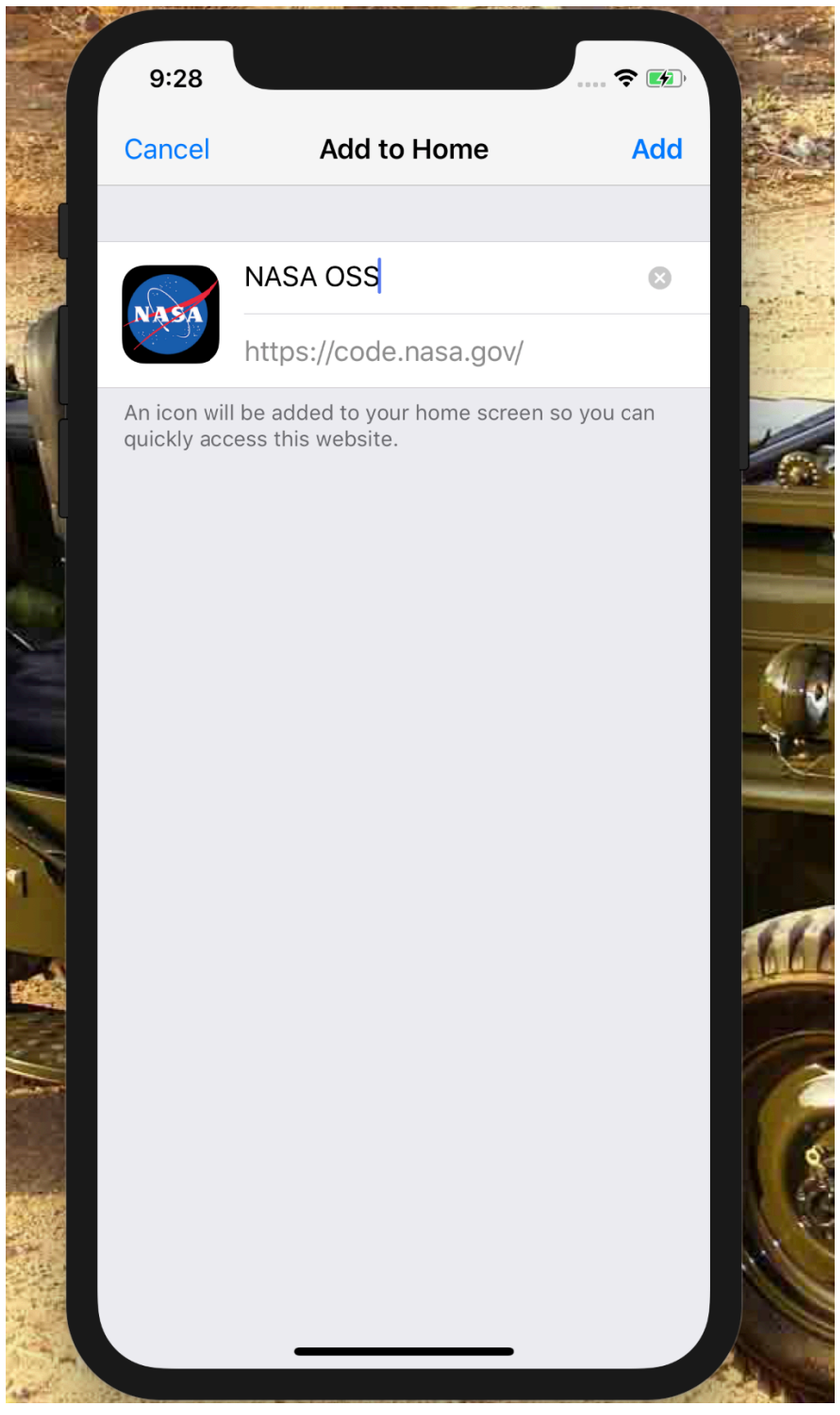
Lyft

Financial Times

..

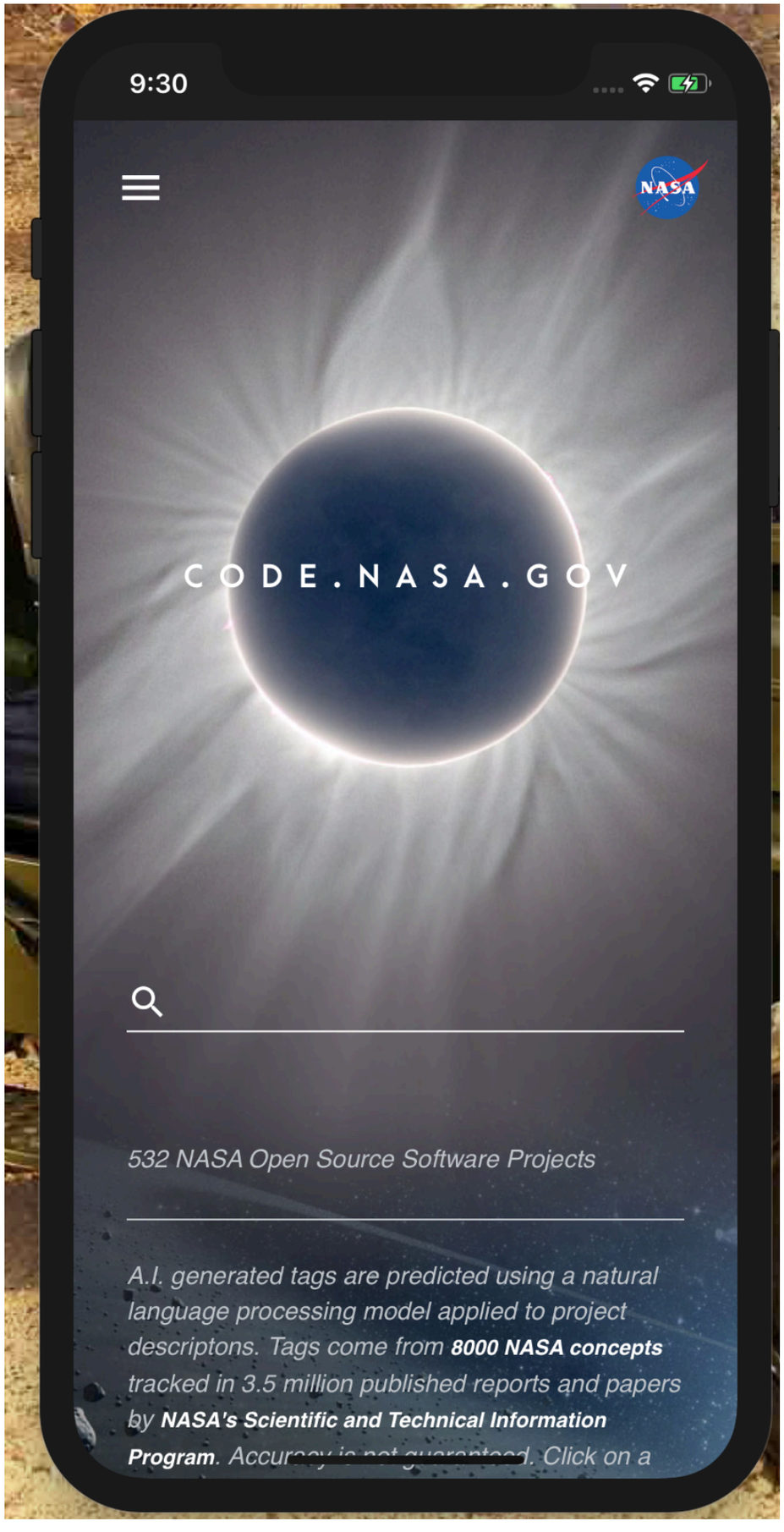
- PWAs


9:27
code.nasa.gov
AirDrop. Share instantly with people nearby. If they turn on AirDrop from Control Center on iOS or from Finder on the Mac, you'll see their names here. Just tap to share.
Reminders
More
Open in News
Add to Home Screen
Print
Request Desktop Site
Cancel


9:28
Cancel
Add to Home
Add
NASA OSS
https://code.nasa.gov/
An icon will be added to your home screen so you can quickly access this website.


9:29
Watch
Files
Extras
NASA OSS

- PWAs


9:30
NASA
CODE.NASA.GOV
532 NASA Open Source Software Projects
A.I. generated tags are predicted using a natural language processing model applied to project descriptons. Tags come from **8000 NASA concepts** tracked in 3.5 million published reports and papers by **NASA's Scientific and Technical Information Program**. Accuracy is not guaranteed. Click on a

- PWAs

Architecture

- create a fast site
  - HTML
  - client/server rendered
  - AMP
- app shell caching
  - app shell renders the site using a group of resources
  - service workers API
  - enables faster loading of the app
  - offline too
- install
  - WAM
  - full screen
  - home screen icon

- PWAs

  Architecture

    - integrate with os/browser

        - push
        - sensors, & other h/ws
        - payment
        - native app integration

- PWAs
  - Service Workers
    - browser script
    - runs in the background
    - outside of web page
    - push support
    - background sync
    - network request interceptions
    - programmatic app's cache management of responses
    - web app install banners (along with WAM) //changing spec.
    - cache and fetch resources
      - resource request interception to provide local files
      - offline
  - JS file with app. like features required by the PWA

- PWAs
  - Service Workers Life Cycle
    - register SW with the browser

      on home page load

      register in the correct scope of the PWA

      Navigator.serviceWorker.register()

    - listen and handle the install event

      install event is fired once

      PWA interaction

      offline cache (along with storage API)

      promise notification

- PWAs
  - Service Workers Life Cycle
    - listen and handle the activate event

      navigation events like fetch can be handled by SW if its installed & active

      replace/update cached files

      claim() on client's object

        4 page fetch to go via a SW

      skipWaiting()

  - listen and handle the fetch events

      navigation events can be interrupted to provide custom functionality

      offline behaviour

      native app like

- PWAs
  - Service Workers Life Cycle
    - listen and handle the before install prompt event

      chrome 67 & earlier (shows “add to home screen if”)

      - app is not already installed
      - 30 sec user interaction
      - secure app
      - a SW is registered having a fetch event handler
      - WAM with a name/shot_name property
      - WAM with min. two icons (512px & 192px)
      - WAM with a start_url prop.
      - WAM with a display prop. of standalone/ fullscreen/ minimal-ui

- PWAs
  - Service Workers Life Cycle
    - listen and handle the before install prompt event
      chrome 68 & above
      - listen for BeforeInstallPrompt
      - user notification to install the app
      - prompt()
      - backwards compatibility
      - install button as needed (future) 

- PWAs
  - Service Workers Life Cycle
    - listen and handle the push event
      - Web Push API mimicking the native messaging feature
      - for app reengagement

- PWAs

Promises

- placeholders that maybe resolve/rejected/

```
var prom = new Promise(function(resolve, reject) {

    //async. task

    if(tasks_done)
        resolve();
    else
         reject();

});
```

- then() method

```
prom.then(function(result) {
    //full-filled prom. handling here
},function(err) {
   //rejected prom. handling here
});
```

- catch() method

```
prom.then(function(result) {
    //full-filled prom. handling here
});

prom.catch(function(err) {
    //rejected prom. handling here
});
```

- PWAs

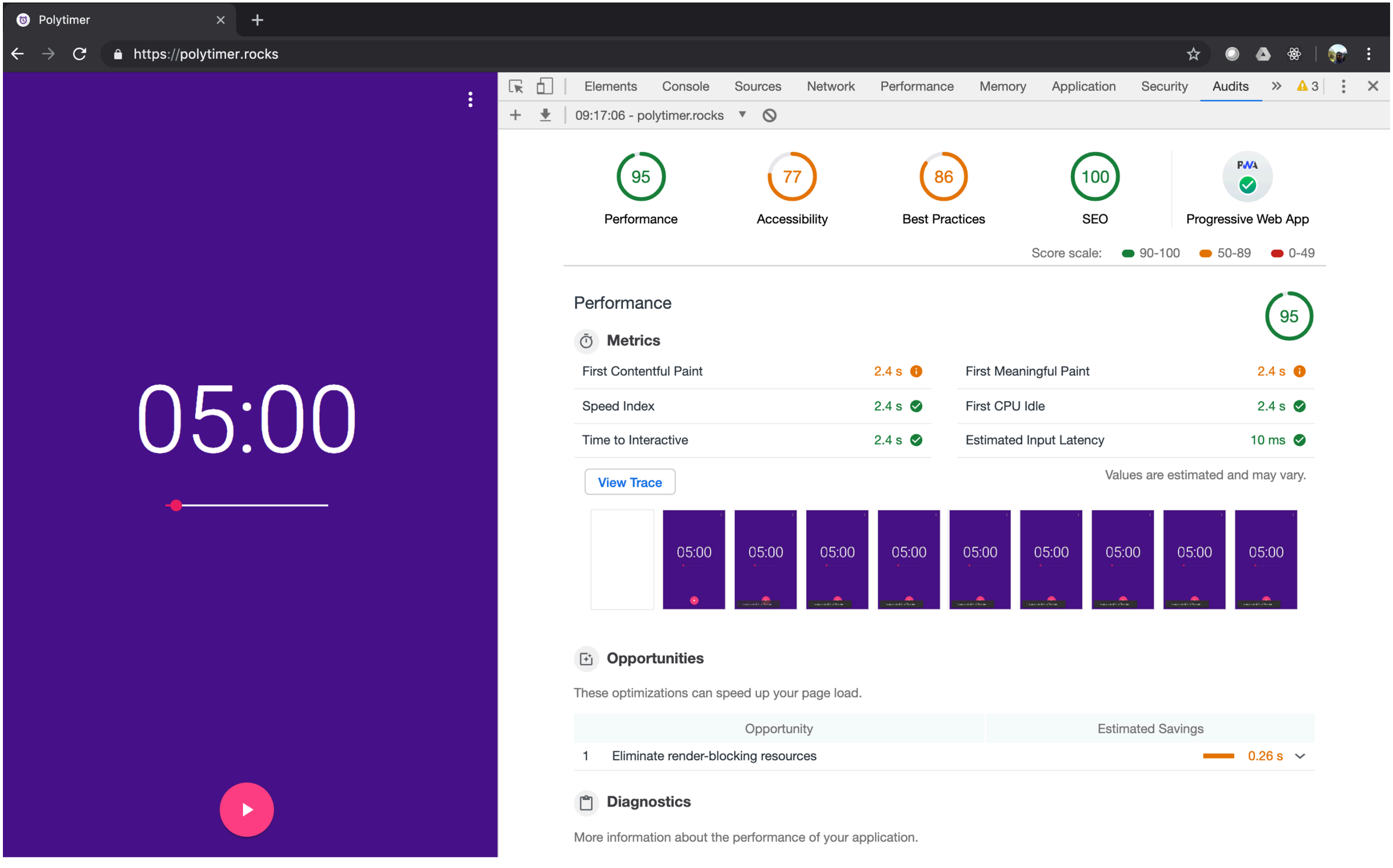
  Promises

    - chain then()

```
prom.then(function(result) {
    //full-filled prom. handling here
}).then(function(result) {
    ////full-filled prom. handling here (when the prev. prom. is done)
});
```

https://developers.google.com/web/fundamentals/primers/promises

- PWAs
  - Auditing
    - Lighthouse https://developers.google.com/web/tools/lighthouse/



- PWAs
    - app icon
        - square, at least 260 x 260 px

          https://realfavicongenerator.net
    - manifest
        - list of contents / checklist (json)
        - web app info to the browser
            - name
            - icons
            - color scheme
            - startURL

https://developers.google.com/web/fundamentals/web-app-manifest/