

- 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

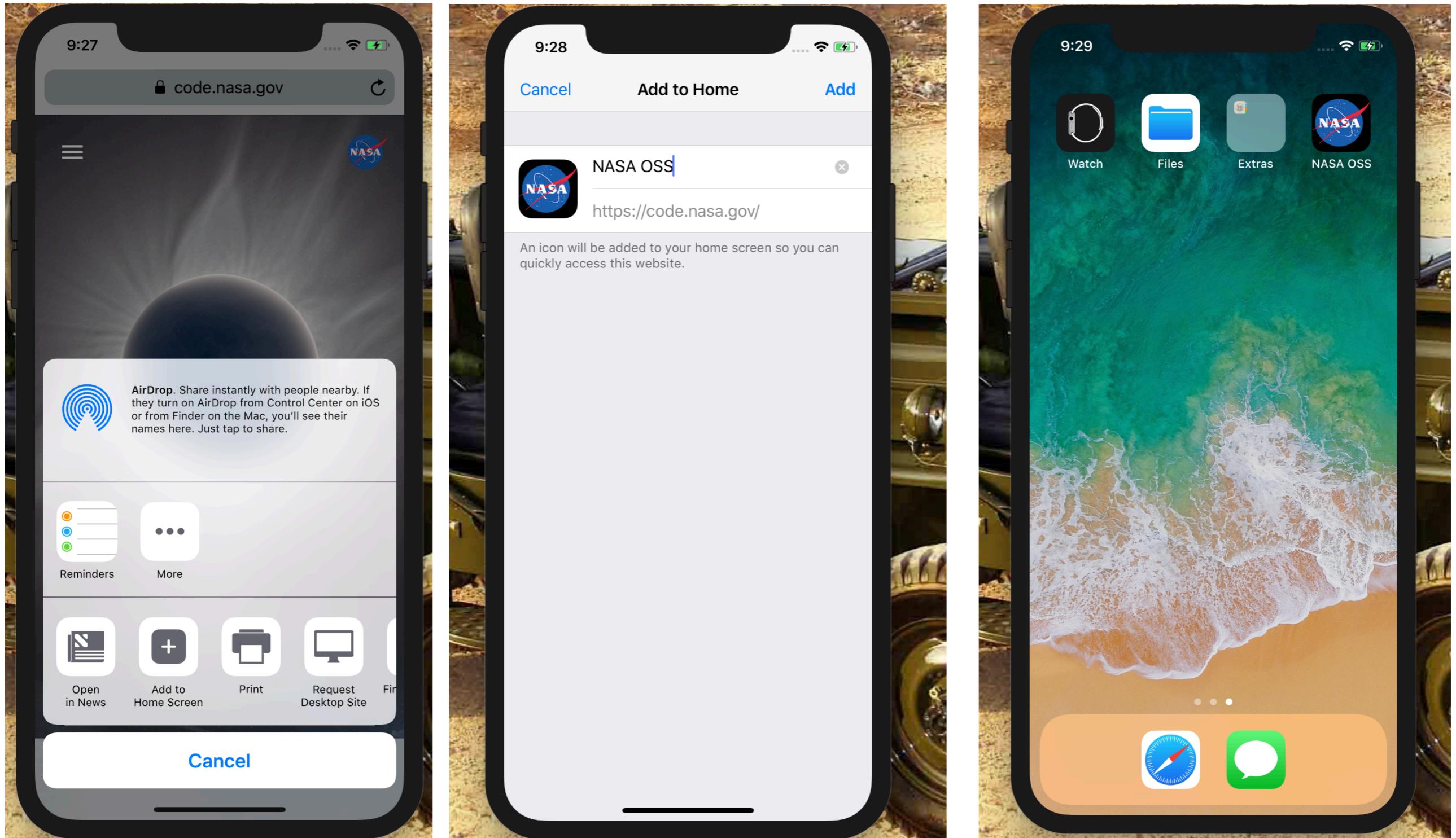
Forbes

Lyft

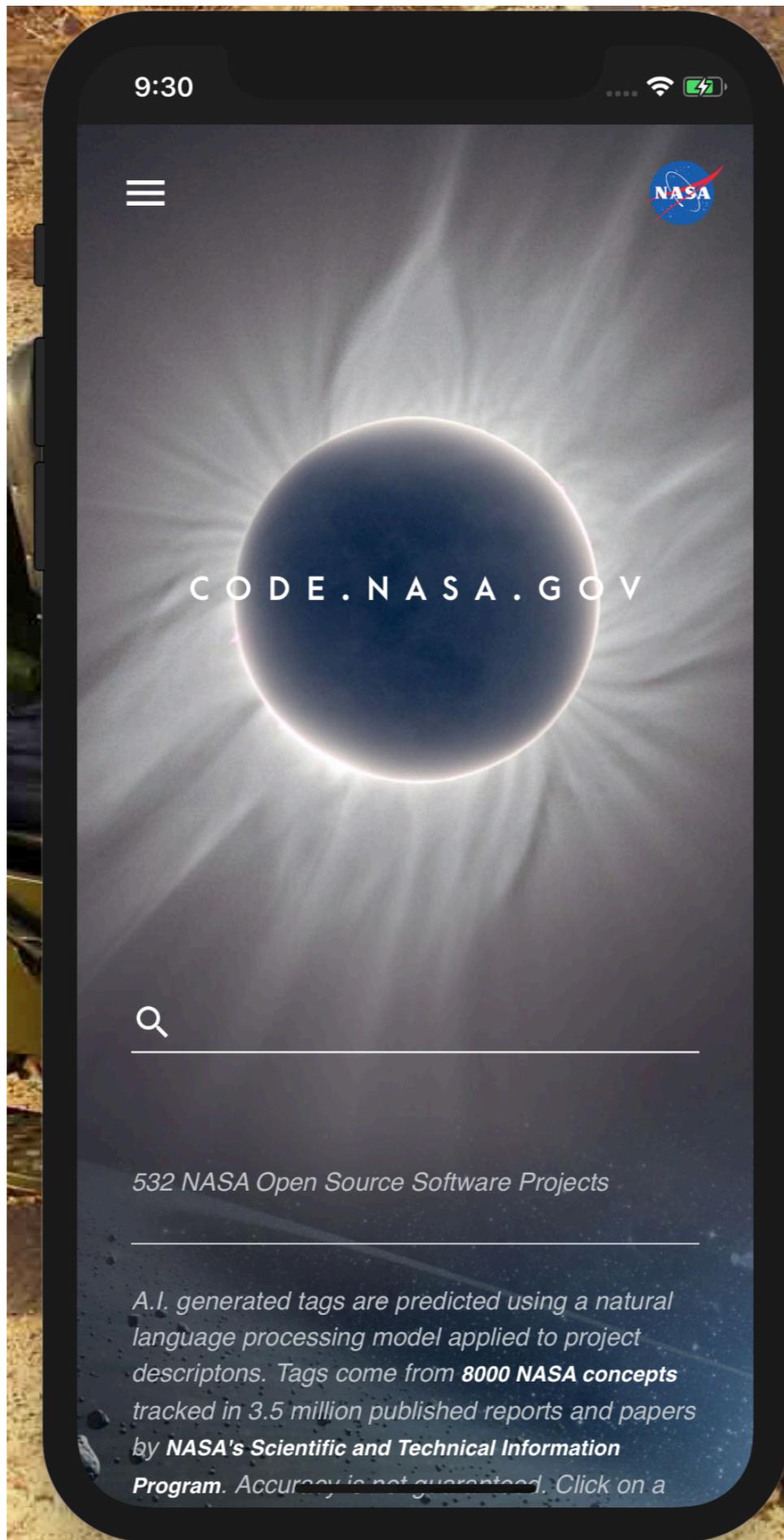
Financial Times

..

- PWAs



- PWAs



- 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/w/s
 - 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
 - <https://caniuse.com>

- PWAs
- Service Workers Life Cycle (RIAFHP)
 - 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
 - first chance for 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() to remove a old SW & activate new one
 - 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 with the site
 - 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 (btn. click)

- handle prompt() if app is installed

- 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 have actions that are resolved/rejected/pending/settled

```
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/>

The screenshot shows the Lighthouse audit results for the URL <https://polytimer.rocks>. The overall score is 95, categorized under Performance. The audit interface includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. A large green circle indicates the PWA status. The Performance section details metrics like First Contentful Paint, Speed Index, and Time to Interactive, all achieving 2.4s. Opportunities for optimization are listed, such as eliminating render-blocking resources which could save 0.26s. Diagnostics provide more information about the application's performance.

05:00

95

Performance

Metrics

Metric	Value	Status
First Contentful Paint	2.4 s	ⓘ
Speed Index	2.4 s	✓
Time to Interactive	2.4 s	✓
First Meaningful Paint	2.4 s	ⓘ
First CPU Idle	2.4 s	✓
Estimated Input Latency	10 ms	✓

View Trace

Values are estimated and may vary.

Opportunities

These optimizations can speed up your page load.

Opportunity	Estimated Savings
1 Eliminate render-blocking resources	0.26 s

Diagnostics

More information about the performance of your application.

- 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/>