HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Progressive web apps

The web, today

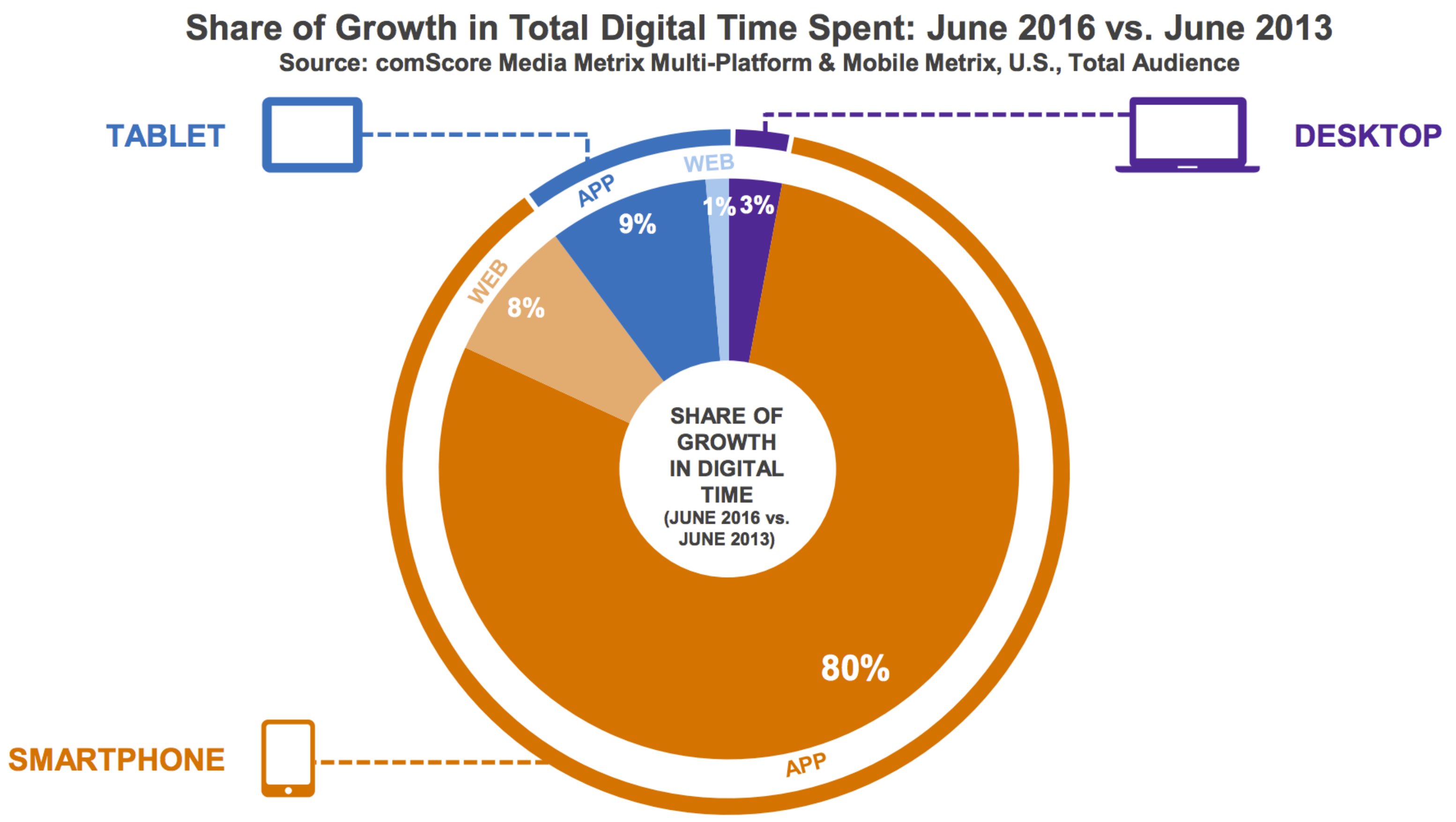
What are the challenges we are facing today with native and web apps ?

Challenges

- **Internet speed** - 60% of the world’s population is still using 2G internet
- **Slow website load** - 53% of users will abandon a site if it takes longer than 3 seconds to load.
- **High friction** - An average user installs 0 applications in a month.
- **User engagement** — Users spend most of their time in native apps, but mobile web reach is almost three times that of native apps

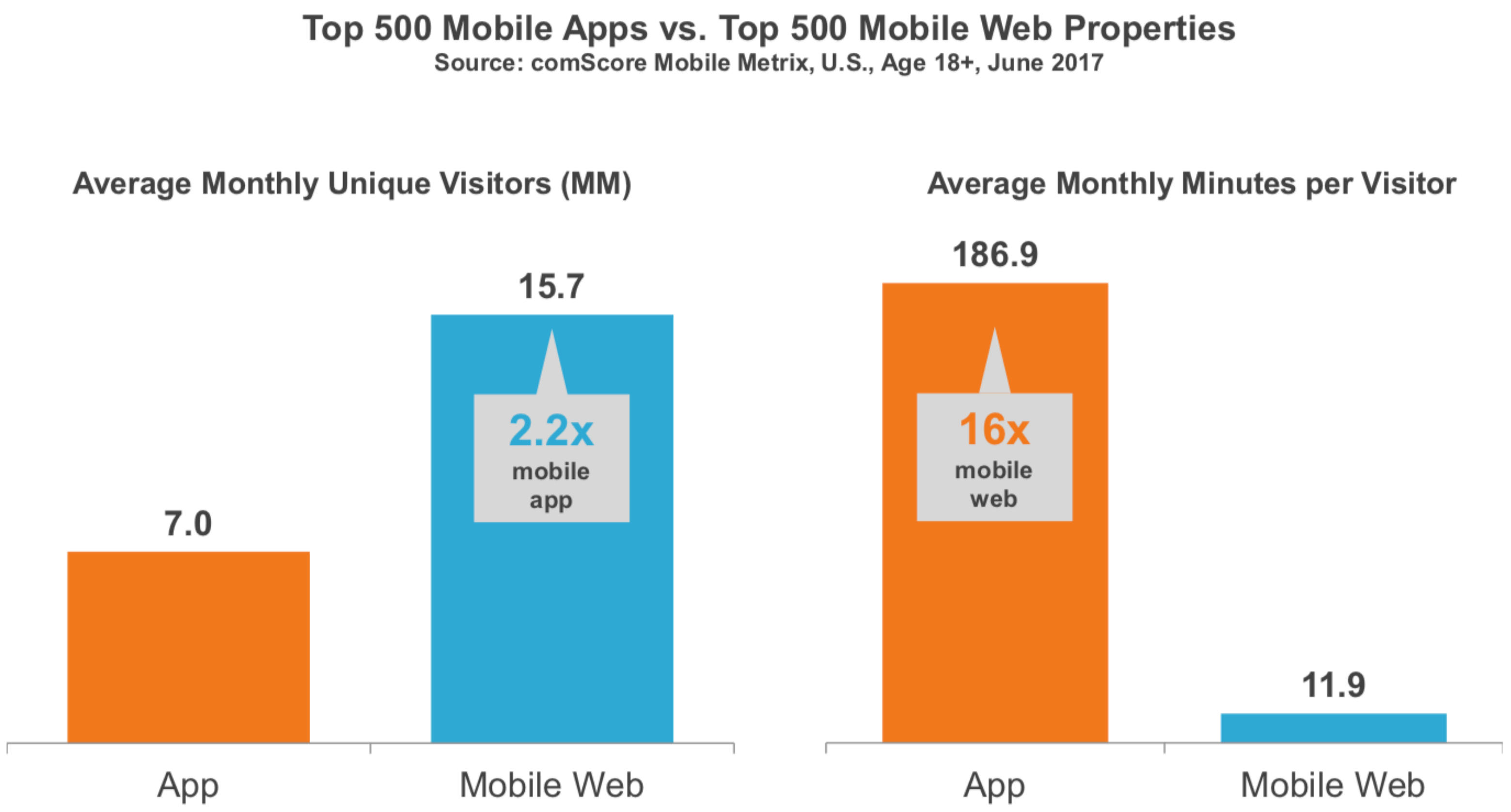
heig-vd

Web vs Mobile



https://www.comscore.com/

Web vs Mobile



https://www.comscore.com/

Progressive web apps, the best of both worlds

#Reliable, #Fast, #Engaging

Reliable

- **Load instantly** and never show the downasaur, even in uncertain network conditions.
- When launched from the user's home screen, **service workers** enable a Progressive Web App to load instantly, regardless of the network state.
- A service worker, written in JavaScript, is like a **client-side proxy** and puts you in control of the cache and how to respond to resource requests
- By **pre-caching** key resources you can eliminate the dependence on the network, ensuring an instant and reliable experience for your users.

Engaging

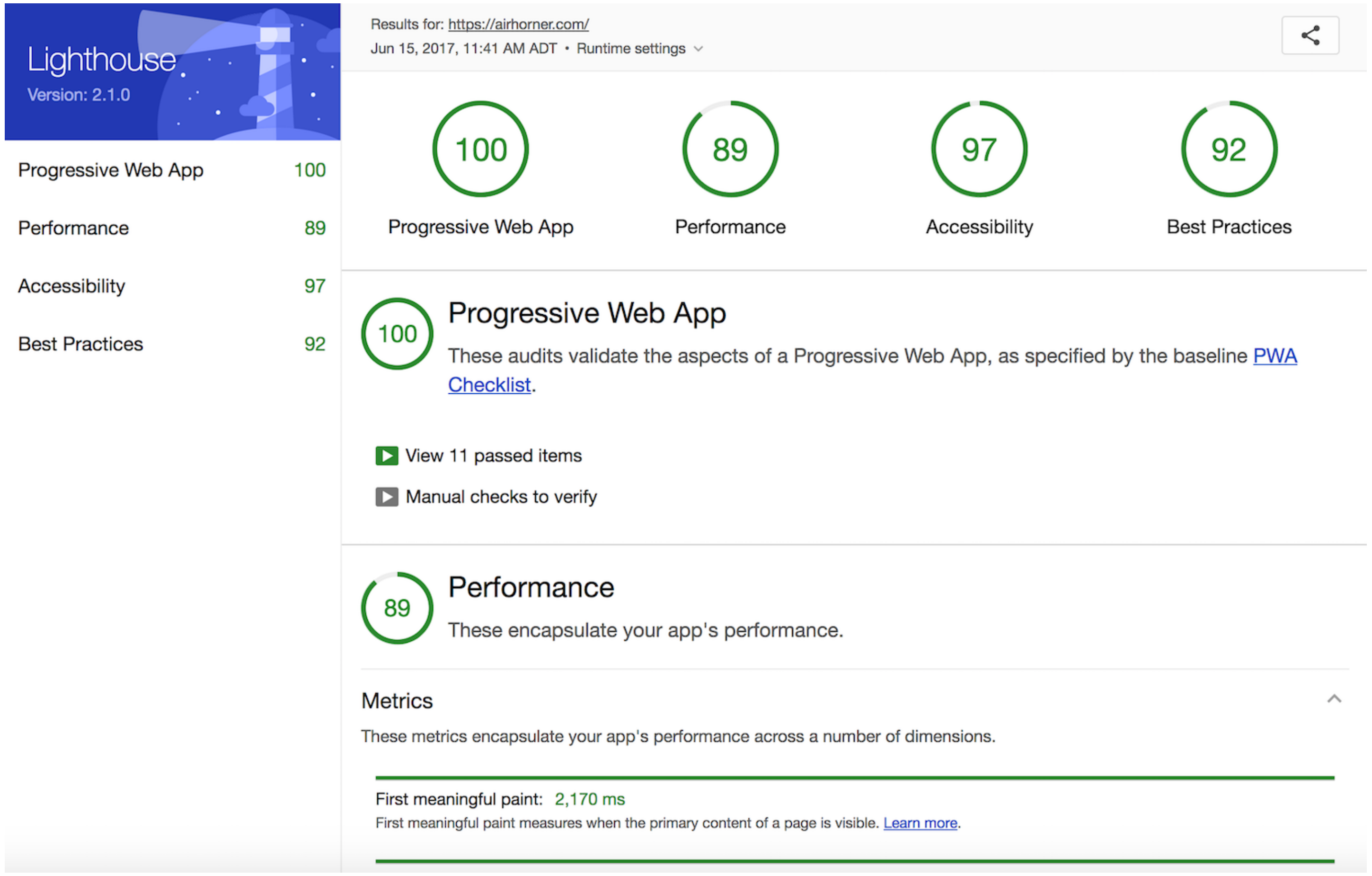
- Progressive Web Apps are **installable** and live on the user's home screen, without the need for an app store
- Can re-engage users with web push **notifications**.
- The **Web App Manifest** allows you to control how your app appears and how it's launched
- You can specify home screen **icons**, the page to load when the app is launched
- You can specify whether or not to show the **browser chrome**.

Fast

- PWAs provide experiences that are **consistently fast**
- The first time, Loads in less than **3 secondes**
- Once loaded, users expect them to be fast – no *janky* scrolling or *slow-to-respond* interfaces.

Lighthouse

Audit for performance, accessibility, progressive web apps, and more.


Lighthouse
Version: 2.1.0

Progressive Web App	100
Performance	89
Accessibility	97
Best Practices	92

Results for: https://airhorner.com/
Jun 15, 2017, 11:41 AM ADT • Runtime settings

100 Progressive Web App
89 Performance
97 Accessibility
92 Best Practices

100 Progressive Web App
These audits validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.

View 11 passed items
Manual checks to verify

89 Performance
These encapsulate your app's performance.

Metrics
These metrics encapsulate your app's performance across a number of dimensions.

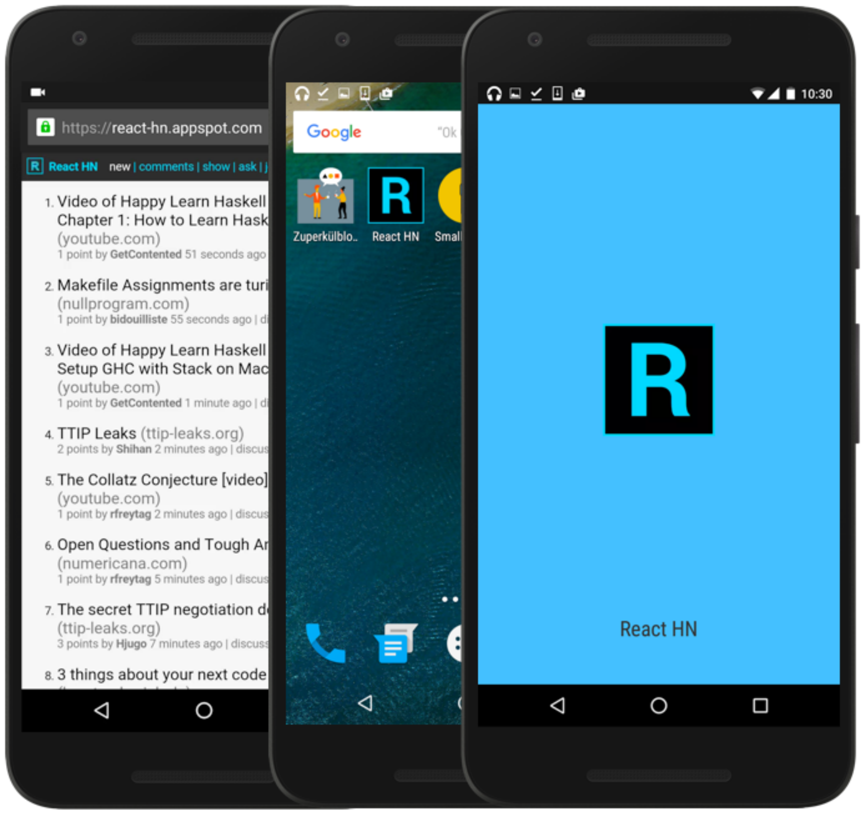
First meaningful paint: 2,170 ms
First meaningful paint measures when the primary content of a page is visible. Learn more.

https://developers.google.com/web/tools/lighthouse/

Add to home screen

Web App Manifest

```
{
  "name": "React HN",
  "short_name": "React HN",
  "icons": [{
      "src": "img/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },...],
  "start_url": "index.html",
  "background_color": "#4CC1FC",
  "display": "standalone",
  "theme_color": "#222222"
}
```



Features

- User can be prompted to Add to Homescreen
- The app is loaded with a splash screen
- The theme is customized to match the brand colors

How it works

Create a `manifest.json` file for a progressive web app.

```
{
  "name": "React HN",
  "short_name": "React HN",
  "icons": [{
      "src": "img/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    }, {
      "src": "img/splashscreen-icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
  }],
  "start_url": "./?utm_source=web_app_manifest",
  "background_color": "#4CC1FC",
  "display": "standalone",
  "theme_color": "#222222"
}
```

How it works

When you have created the manifest, add a link tag in the `head` of your page to reference it:

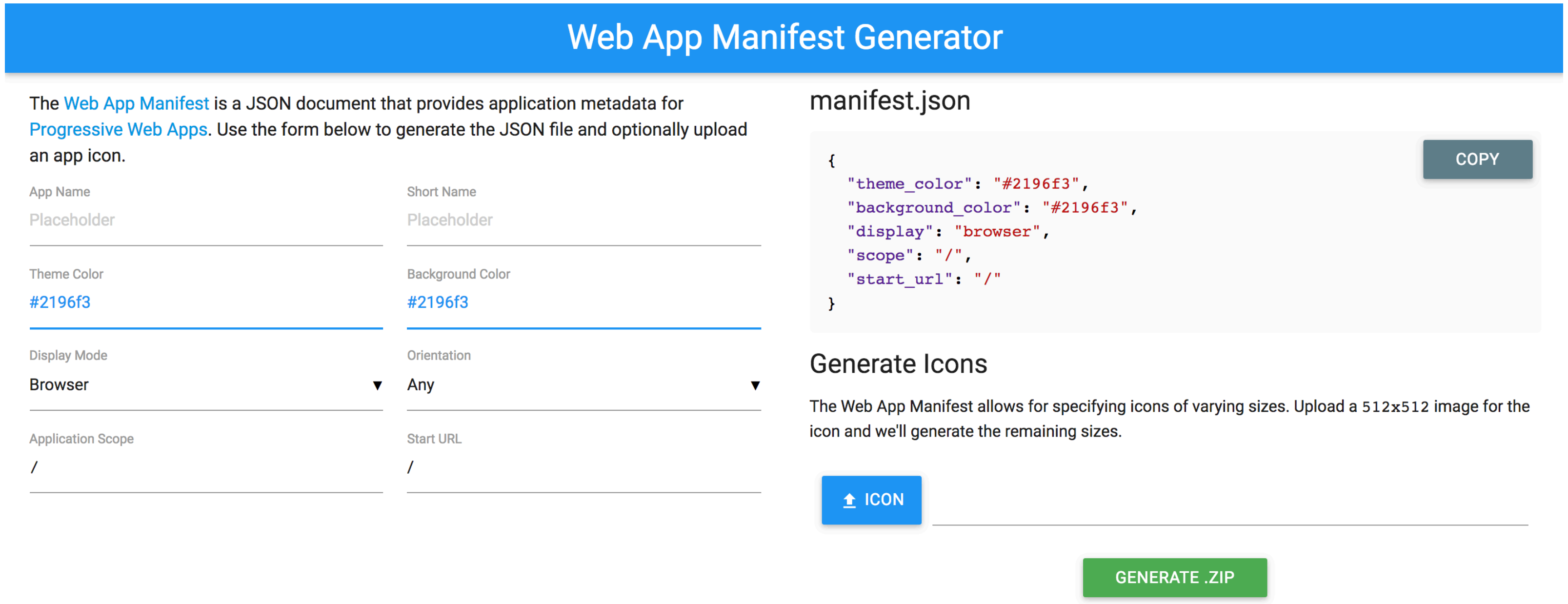
```
<link rel="manifest" href="/manifest.json">
```

That's it !

heig-vd

Add to Home Screen

Manifest generator



Web App Manifest Generator

The Web App Manifest is a JSON document that provides application metadata for Progressive Web Apps. Use the form below to generate the JSON file and optionally upload an app icon.

App Name: Placeholder	Short Name: Placeholder
Theme Color: #2196f3	Background Color: #2196f3
Display Mode: Browser	Orientation: Any
Application Scope: /	Start URL: /

manifest.json

```
{
  "theme_color": "#2196f3",
  "background_color": "#2196f3",
  "display": "browser",
  "scope": "/",
  "start_url": "/"
}
```

Generate Icons

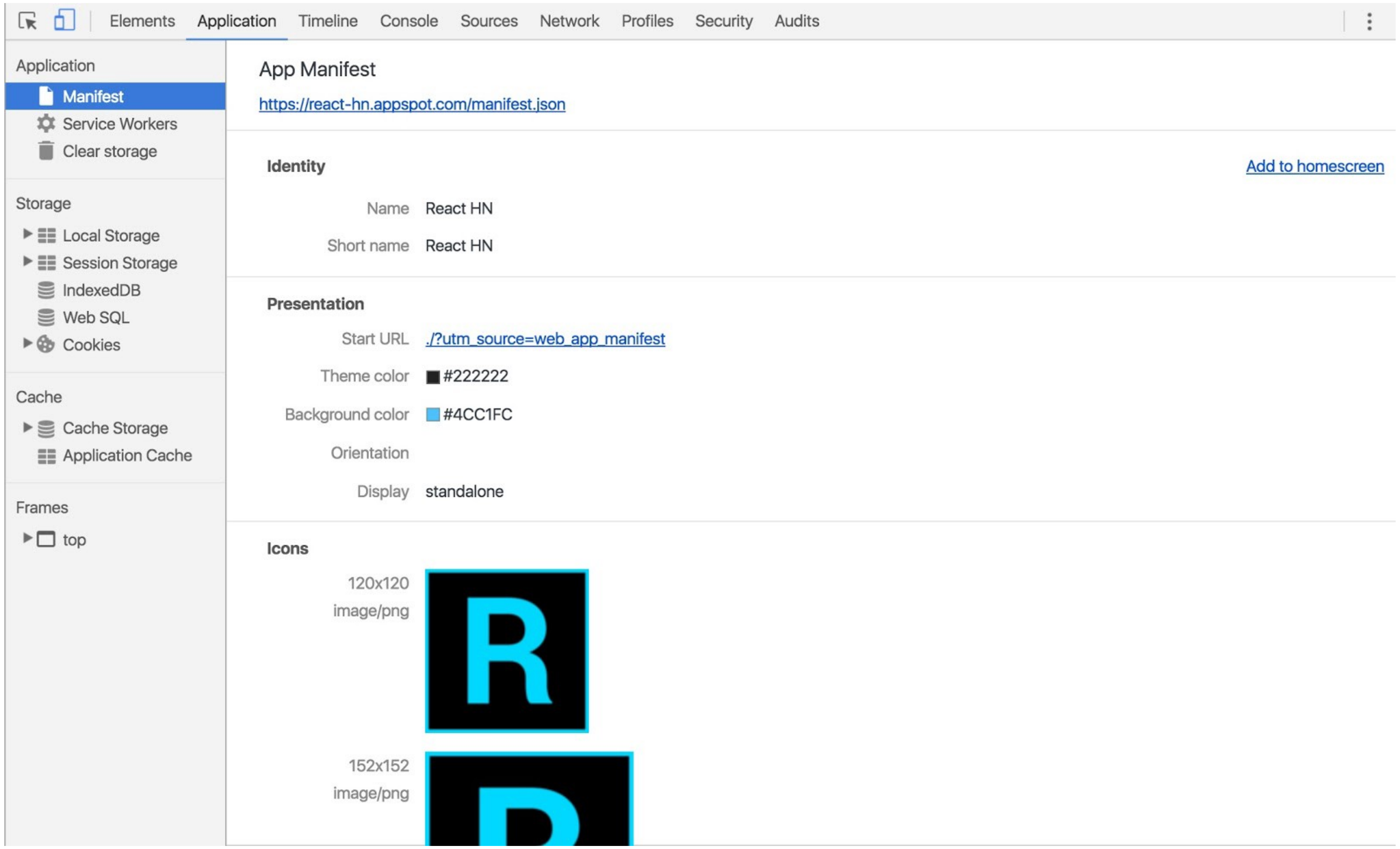
The Web App Manifest allows for specifying icons of varying sizes. Upload a `512x512` image for the icon and we'll generate the remaining sizes.

https://app-manifest.firebaseapp.com/

heig-vd

Add to Home Screen

Debugging



App Manifest

https://react-hn.appspot.com/manifest.json

Identity

Name	React HN
Short name	React HN

Presentation

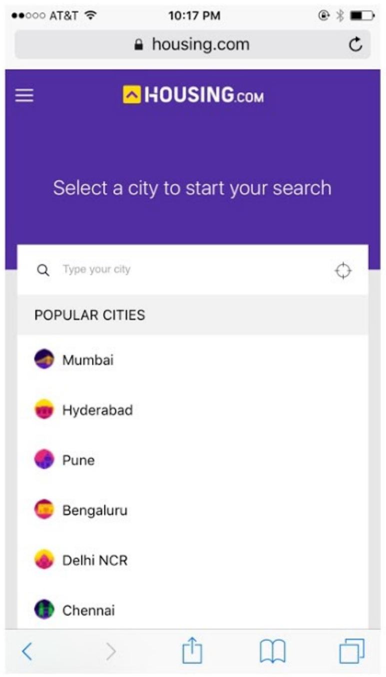
Start URL	./?utm_source=web_app_manifest
Theme color	■ #222222
Background color	🟦 #4CC1FC
Orientation	
Display	standalone

Icons

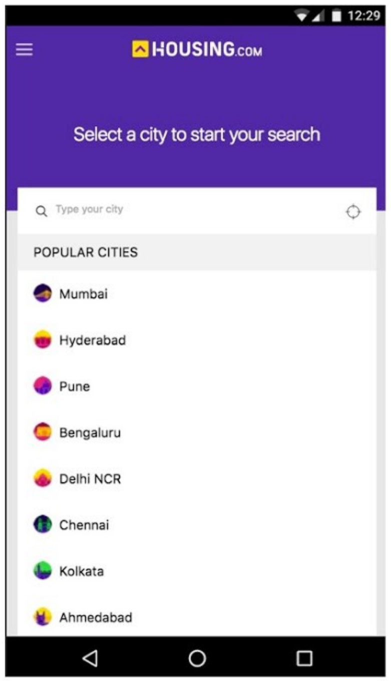
120x120
image/png

152x152
image/png

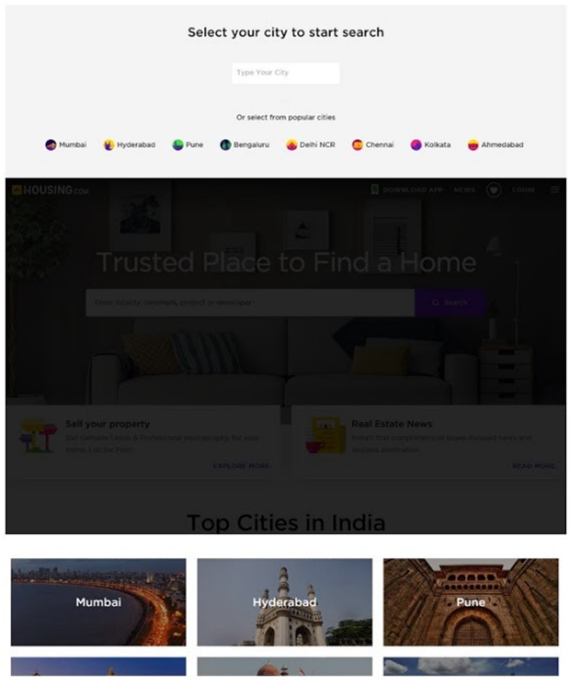
Design is mobile-friendly



Safari on iOS



Chrome on Android



Edge on Windows 10 (Desktop)

https://housing.com

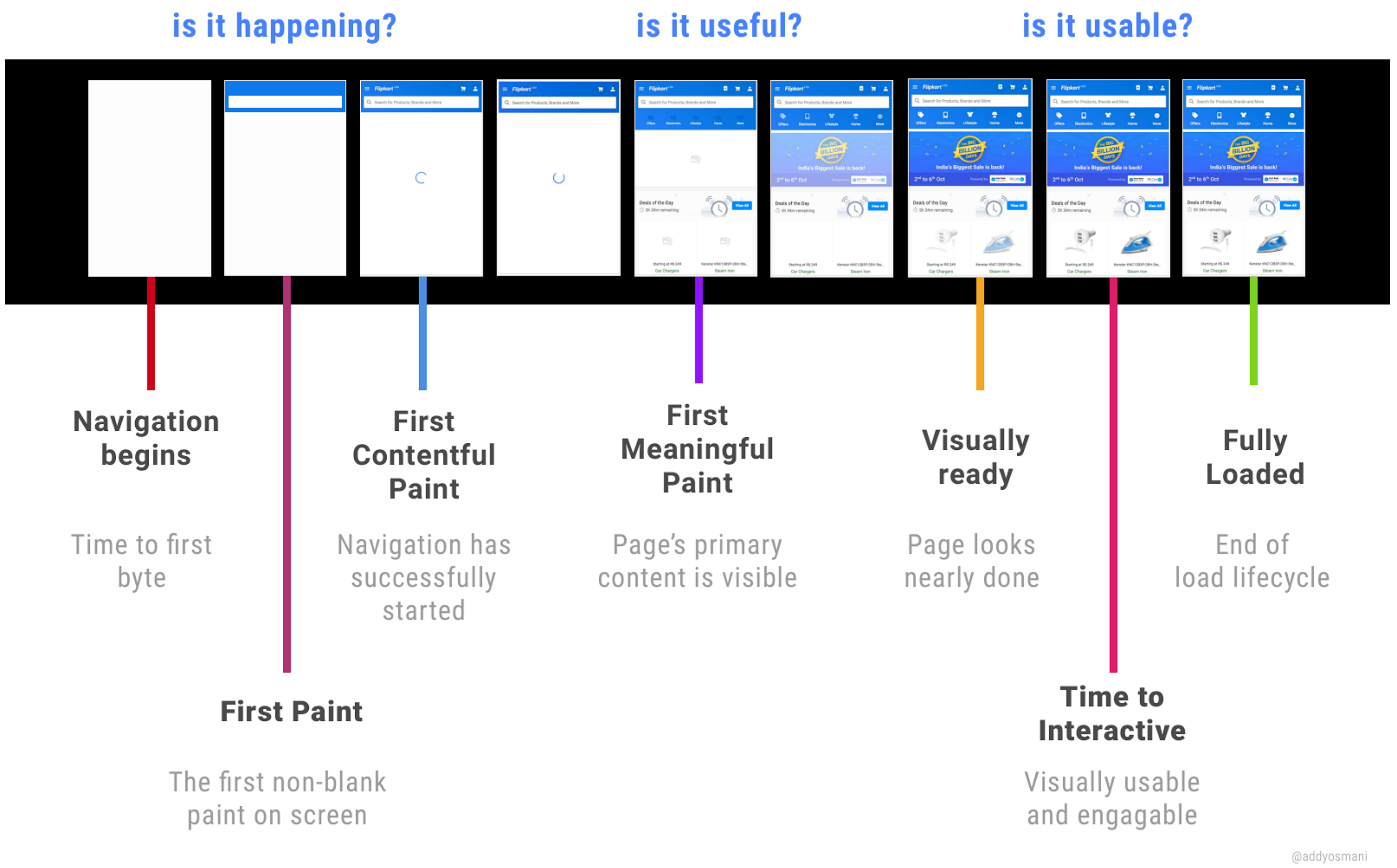
How ?

- Apps optimized for multiple devices should include a meta-viewport in the of their document:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- Adapt the layout depending on the device width
    - with css media queries
    - by sniffing the userAgent on the server
    - or using different urls for mobile/desktop layout

Ensure page load is fast



https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-2-page-load-performance-33b932d97cf2

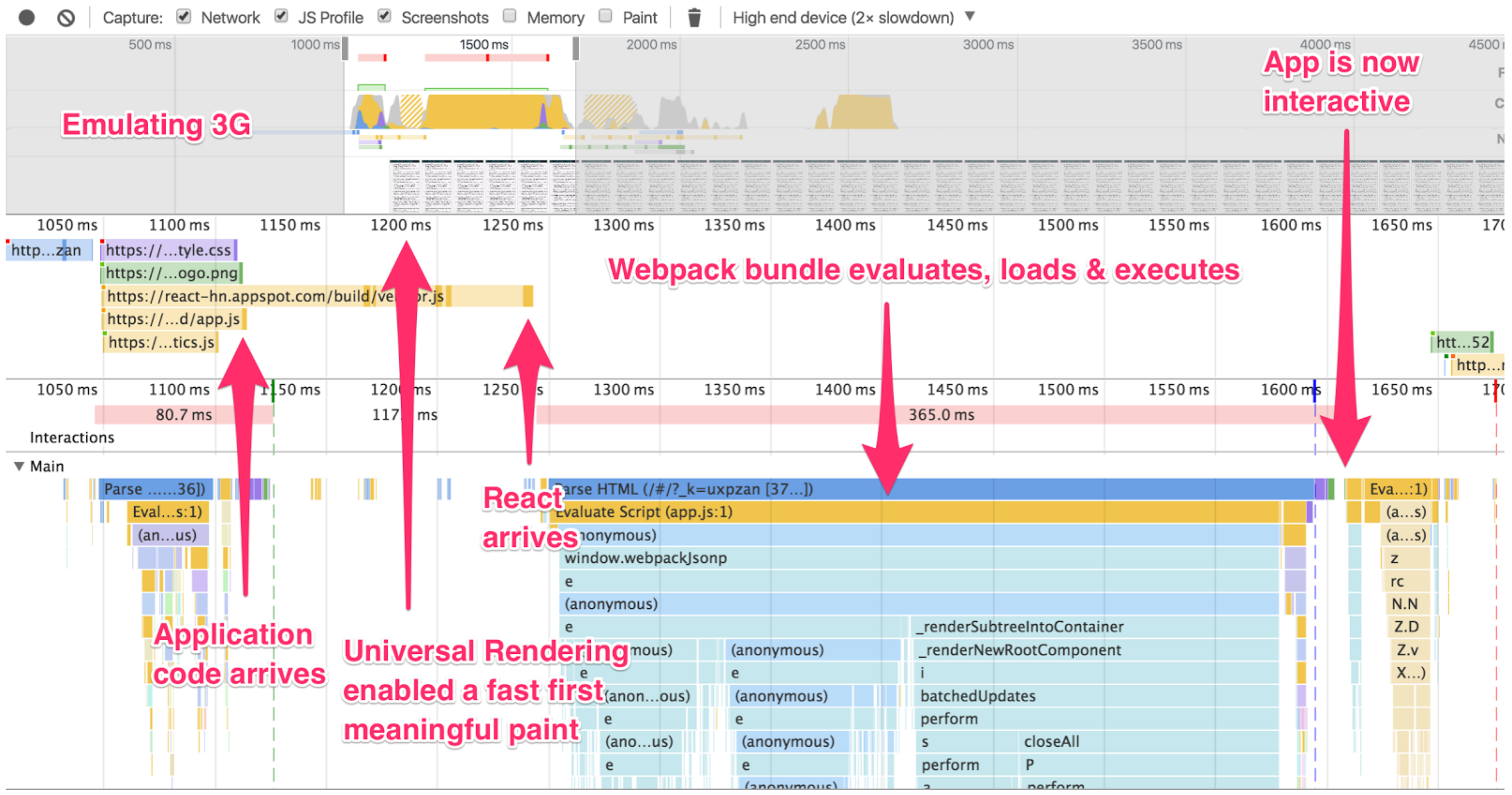
Key performance indicator

- First meaningful paint - when is the main content of the page visible
- Speed Index - visual completeness
- Estimated Input Latency - when is the main thread available to immediately handle user input
- Time To Interactive - how soon is the app usable & engagable

> Goals: Be interactive in < 5s on first visit & < 2s on repeat visits once a Service Worker is active.

Ensure page load is fast

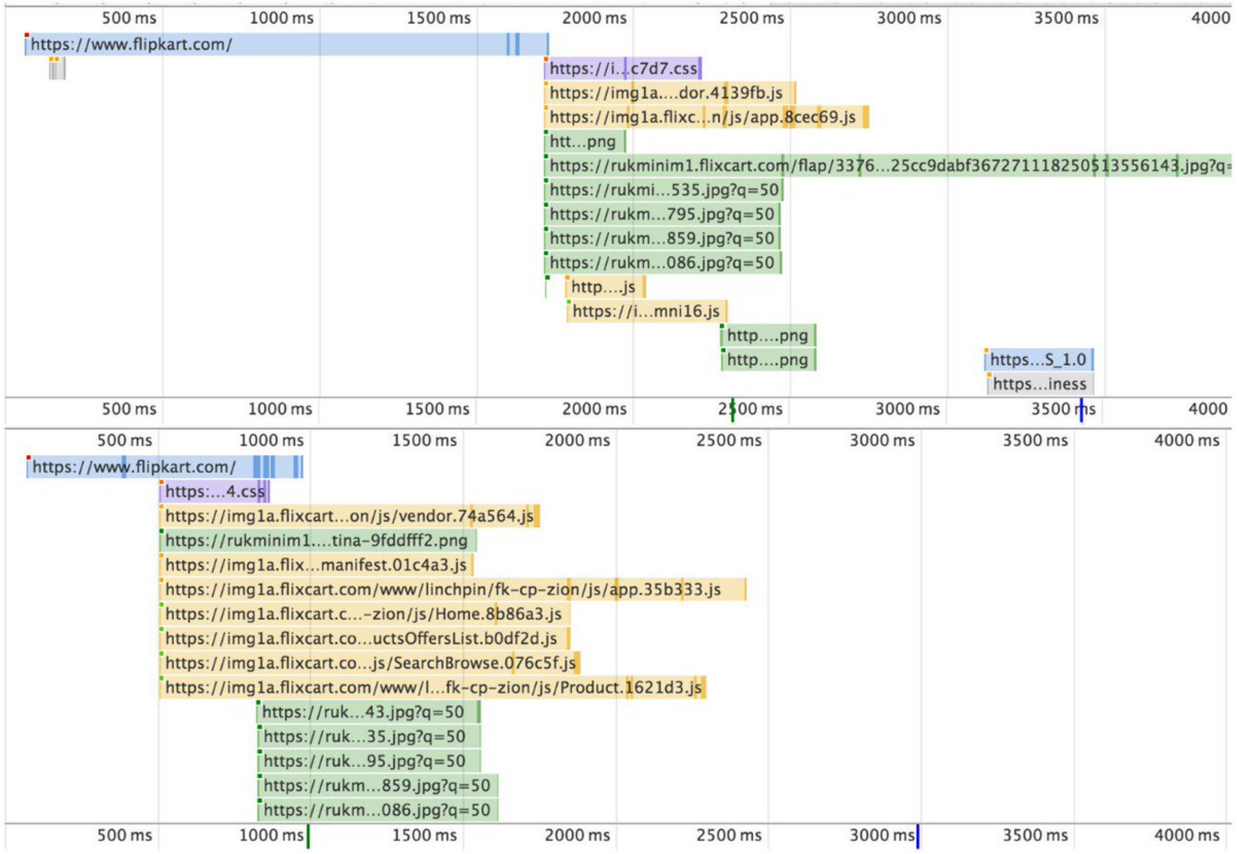
Performance timeline


Capture: Network JS Profile Screenshots Memory Paint
High end device (2× slowdown)
Emulating 3G
App is now interactive
Webpack bundle evaluates, loads & executes
https://...tyle.css
https://...ogo.png
https://react-hn.appspot.com/build/vendor.js
https://...d/app.js
https:/...tics.js
80.7 ms
365.0 ms
Interactions
Main
Parse HTML (/#/?_k=uxpzan [37...])
Evaluate Script (app.js:1)
window.webpackJsonp
_renderSubtreeIntoContainer
_renderNewRootComponent
batchedUpdates
perform
closeAll
Application code arrives
Universal Rendering enabled a fast first meaningful paint
React arrives

Ensure page load is fast

Optimizations @Flipkart

In case you missed this one, here is the before and after of our timeline! pic.twitter.com/XFfRlM6xTg

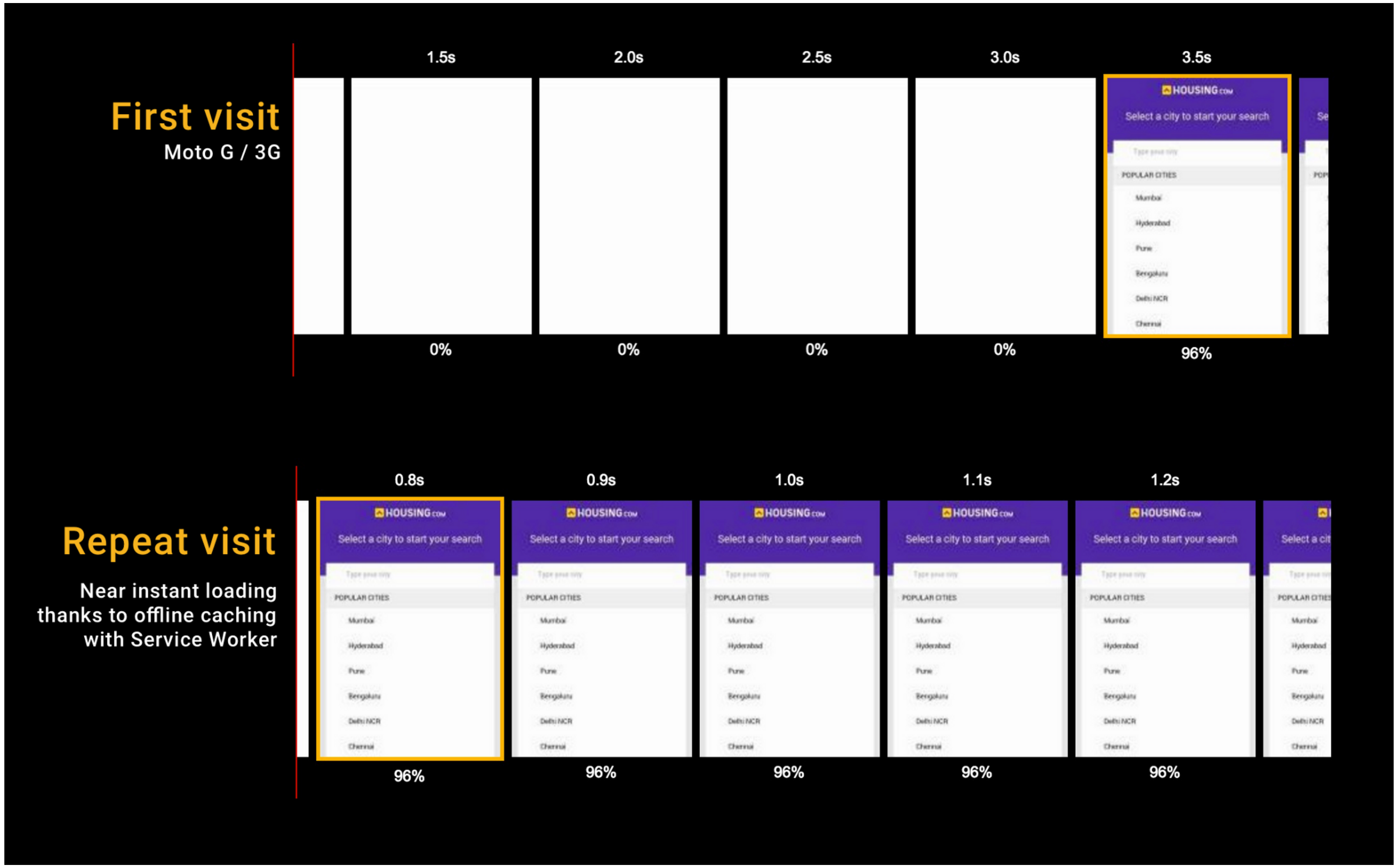


500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000
https://www.flipkart.com/
https://i...c7d7.css
https://img1a....dor.4139fb.js
https://img1a.flixc...n/js/app.8cec69.js
htt...png
https://rukminim1.flixcart.com/flap/3376...25cc9dabf367271118250513556143.jpg?q=
https://rukmi...535.jpg?q=50
https://rukm...795.jpg?q=50
https://rukm...859.jpg?q=50
https://rukm...086.jpg?q=50
http....js
https://i...mni16.js
http....png
http....png
https...S_1.0
https...iness
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
https://www.flipkart.com/
https:...4.css
https://img1a.flixcart...on/js/vendor.74a564.js
https://rukminim1....tina-9fddfff2.png
https://img1a.flix...manifest.01c4a3.js
https://img1a.flixcart.com/www/linchpin/fk-cp-zion/js/app.35b333.js
https://img1a.flixcart.c...-zion/js/Home.8b86a3.js
https://img1a.flixcart.co...uctsOffersList.b0df2d.js
https://img1a.flixcart.co...js/SearchBrowse.076c5f.js
https://img1a.flixcart.com/www/l...fk-cp-zion/js/Product.1621d3.js
https://ruk...43.jpg?q=50
https://ruk...35.jpg?q=50
https://ruk...95.jpg?q=50
https://rukm...859.jpg?q=50
https://rukm...086.jpg?q=50

— Abhinav Rastogi (@_abhinavrastogi) September 2, 2016

heig-vd

Load instantly



Service worker 🎉

A service worker is a background worker that acts as a programmable proxy, allowing us to control what happens on a request-by-request basis. We can use it to make (parts of, or even entire) web apps work offline.

It enables nice features:

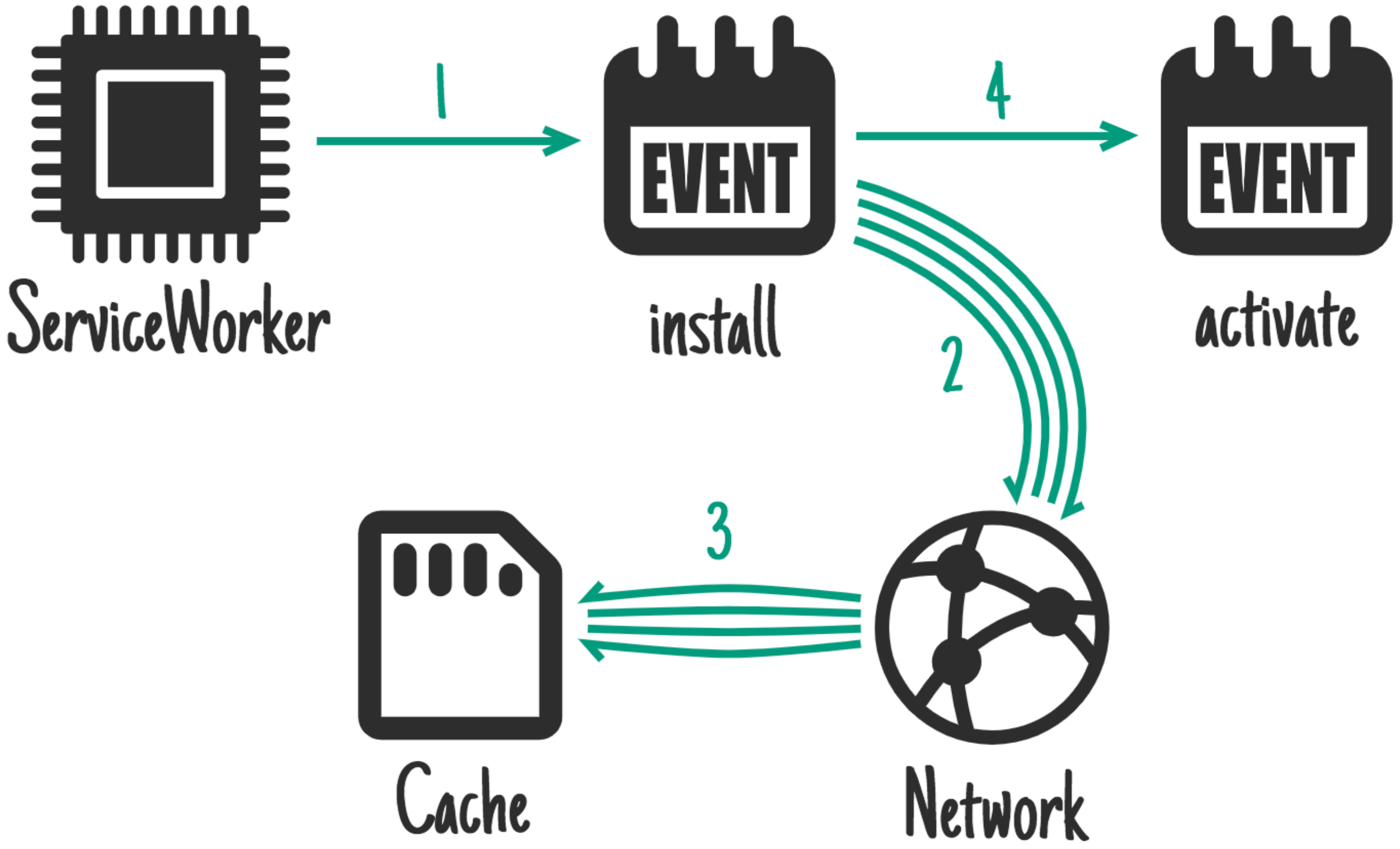
- Push API – A server can send push messages to a web even if the web app or browser are not running
- Background Sync - For deferring actions until the user has stable connectivity

Service worker life

Each service worker goes through three steps in its lifecycle:

- **Registration** – informs the browser where your service worker is located and lets it know it can start installing in the background
- **Installation** – caches the static assets for the page
- **Activation** - starts taking control of the page

Service worker life



Service worker registration

Basic registration in your `index.html` could look like this:

```
<script>
// Check for browser support of service worker
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('service-worker.js')
    .then(function(registration) {
      // Successful registration
    })
    .catch(function(err) {
      // Failed registration, service worker won’t be installed
    });
</script>
```

Service worker installation

```
// service-worker.js
self.addEventListener('install', (event) => {
  event.waitUntil(async function() {
    const cache = await caches.open('mysite-static-v3');
    await cache.addAll([
      '/css/whatever-v3.css',
      '/css/imgs/sprites-v6.png',
      '/css/fonts/whatever-v8.woff',
      '/js/all-min-v4.js'
      // etc
    ]);
  }());
});
```

Service worker activation (intercepting requests)

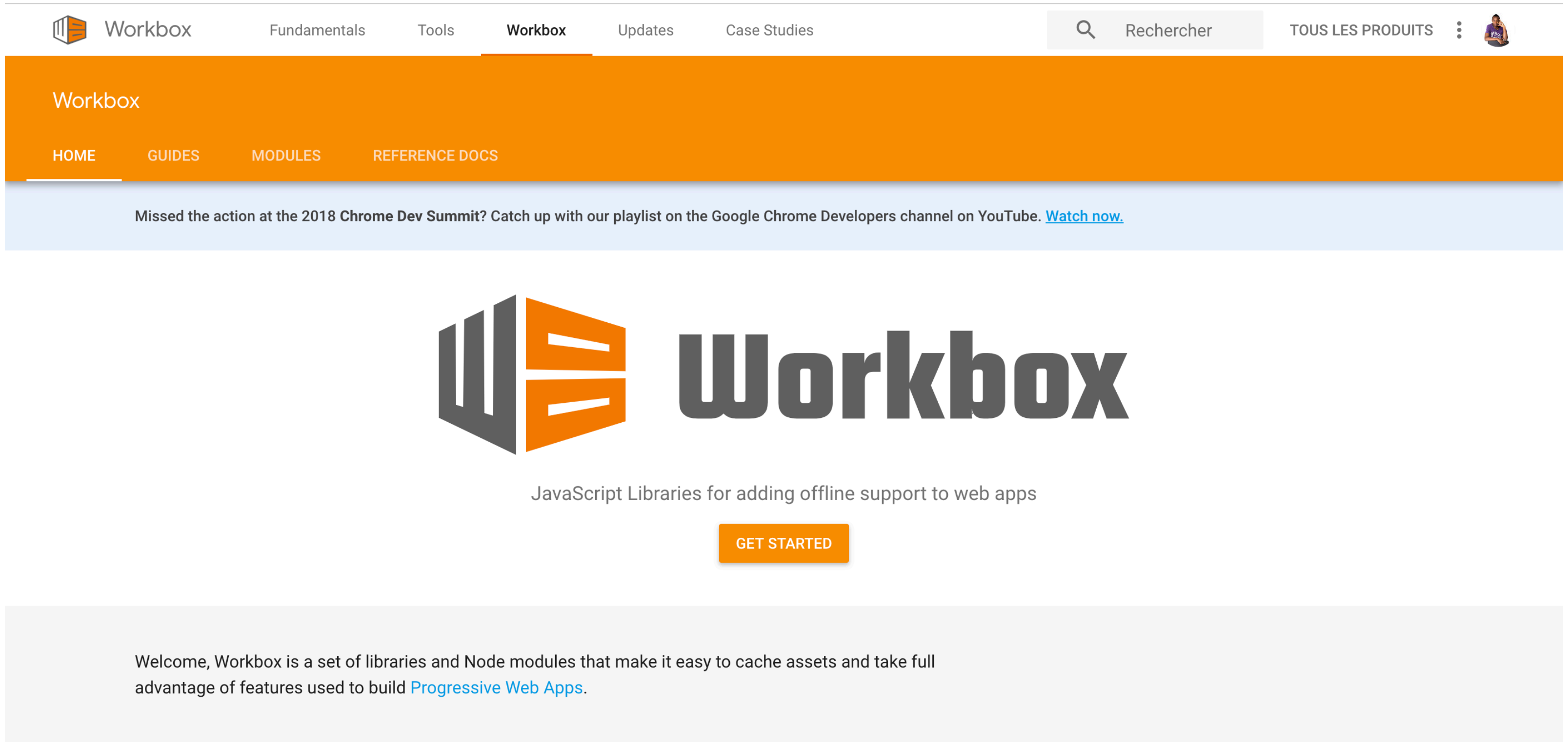
When a service worker controls a page, it can intercept each request being made by the page and decide what to do with it.

```
self.addEventListener('fetch', function(event) {
    console.log(event.request.url);
    event.respondWith(
        caches.match(event.request).then(function(response) {
            return response || fetch(event.request);
        })
    );
});
```

Learn more with examples

heig-vd

Workbox



Workbox Fundamentals Tools Workbox Updates Case Studies Rechercher TOUS LES PRODUITS

Workbox

HOME GUIDES MODULES REFERENCE DOCS

Missed the action at the 2018 **Chrome Dev Summit**? Catch up with our playlist on the Google Chrome Developers channel on YouTube. Watch now.

Workbox

JavaScript Libraries for adding offline support to web apps

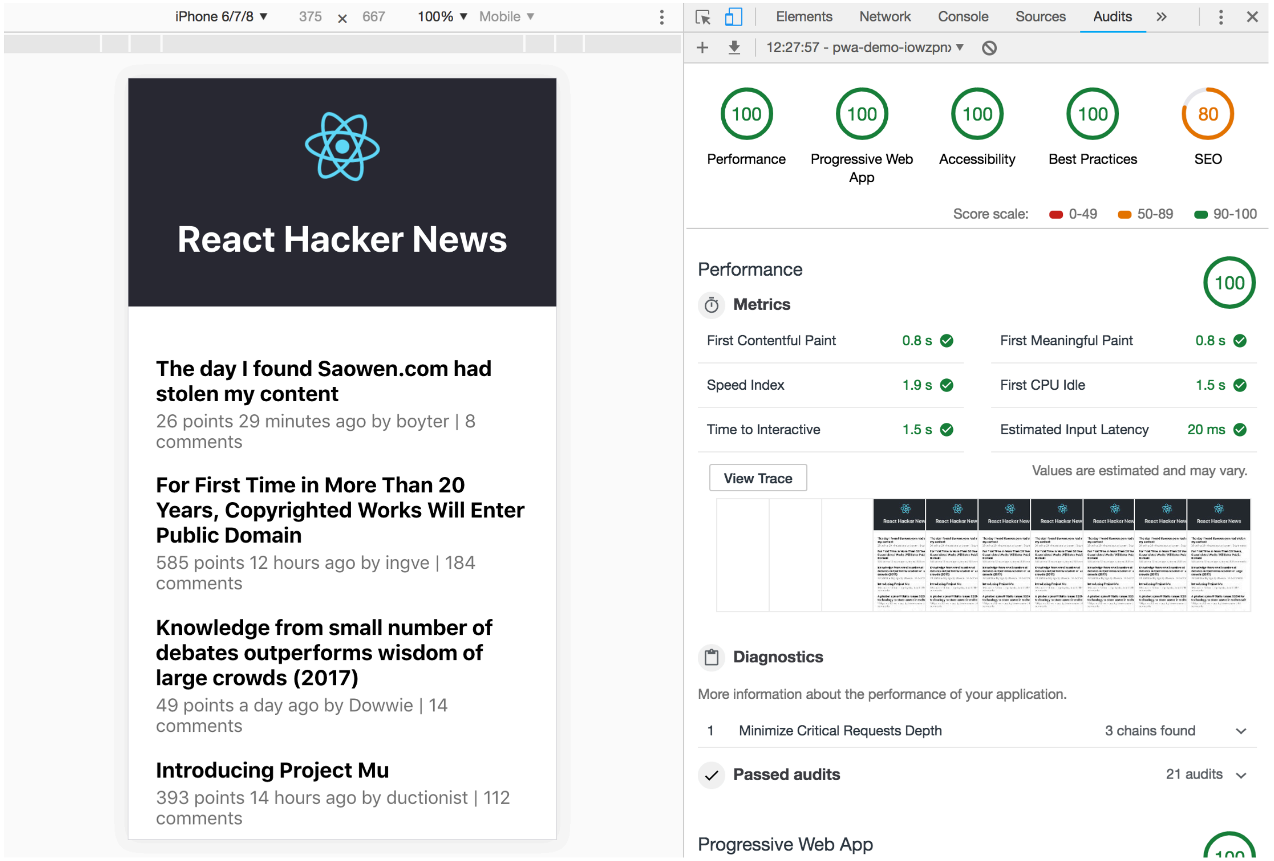
GET STARTED

Welcome, Workbox is a set of libraries and Node modules that make it easy to cache assets and take full advantage of features used to build Progressive Web Apps.

https://developers.google.com/web/tools/workbox/

heig-vd

Progressive web app demo



iPhone 6/7/8 ▾ 375 × 667 100% ▾ Mobile ▾

React Hacker News

The day I found Saowen.com had stolen my content
26 points 29 minutes ago by boyter | 8 comments

For First Time in More Than 20 Years, Copyrighted Works Will Enter Public Domain
585 points 12 hours ago by ingve | 184 comments

Knowledge from small number of debates outperforms wisdom of large crowds (2017)
49 points a day ago by Dowwie | 14 comments

Introducing Project Mu
393 points 14 hours ago by ductionist | 112 comments

Elements Network Console Sources Audits »

12:27:57 - pwa-demo-iowzpn ▾

Performance	Progressive Web App	Accessibility	Best Practices	SEO
100	100	100	100	80

Score scale: 🟥 0-49 🟧 50-89 🟩 90-100

Performance 100

Metrics

First Contentful Paint	0.8 s ✓	First Meaningful Paint	0.8 s ✓
Speed Index	1.9 s ✓	First CPU Idle	1.5 s ✓
Time to Interactive	1.5 s ✓	Estimated Input Latency	20 ms ✓

View Trace

Values are estimated and may vary.

Diagnostics

More information about the performance of your application.

1 Minimize Critical Requests Depth 3 chains found

Passed audits 21 audits

Progressive Web App

Github: https://github.com/heig-vd-tweb/react-pwa

Live demo: https://pwa-demo-iowzpnxasa.now.sh/

Awesome resources

Progressive Web Apps with React.js by Addy Osmani

- Introduction
- Performance
- Offline and network resilience
- Progressive enhancement
- A React and Preact Case PWA Study