Advanced Progressive Web Apps

INSTALLING THE APPLICATION

Maximiliano Firtman
MOBILE+WEB DEVELOPER
@firt firt.mobi

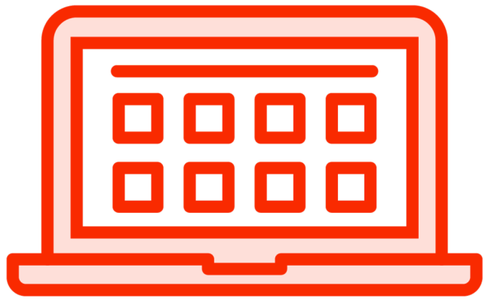
Overview

Installing the Application

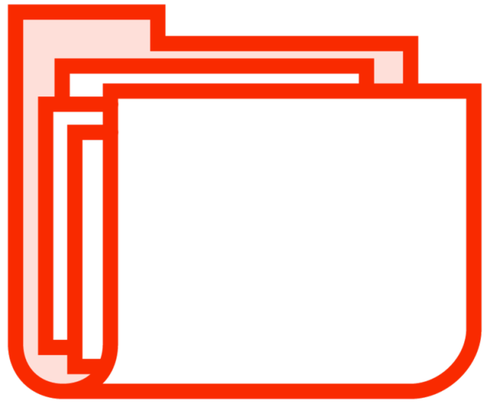
- Installation Architecture Review
- Track Installation for Analytics
- Promote Installation from UI
- Prioritize Store App over Browser Installation
- Improve Reliability on iOS and iPadOS

Installation Architecture

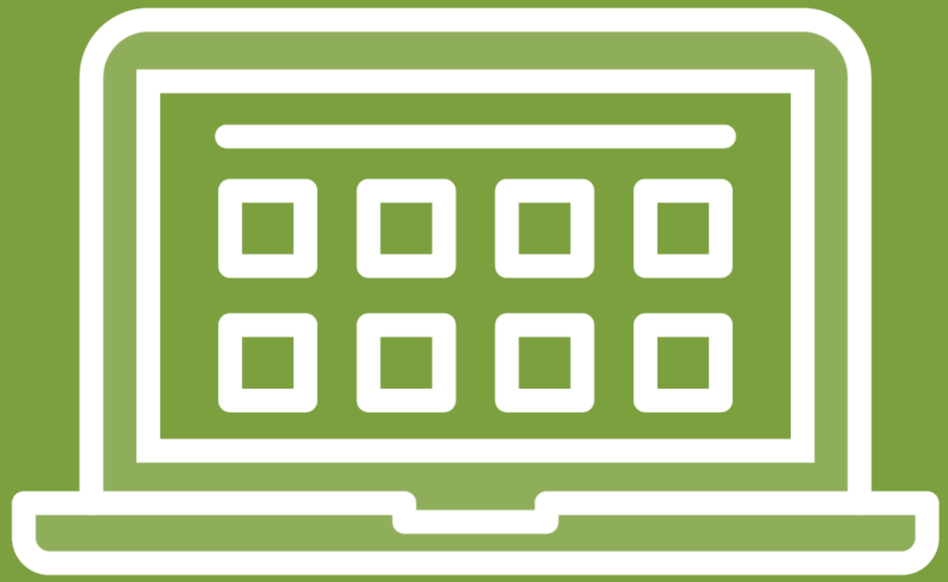
Installation Steps for a PWA



Launcher Icon



App Assets

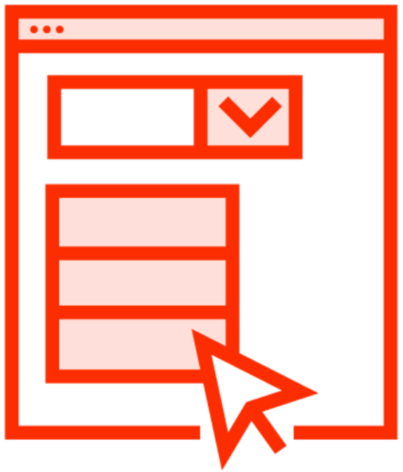


Launcher Icon Installation

It creates a new entry in the launcher menu, start screen or home screen as any other native app on that OS

Installing the launcher icon is what is typically known as installing the app

Launcher Icon Installation

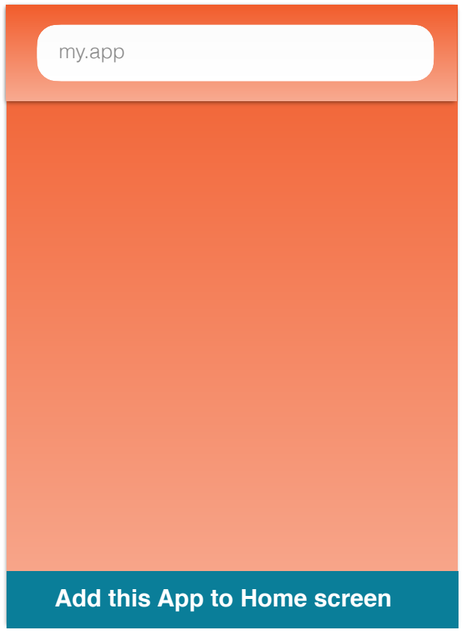


Browser

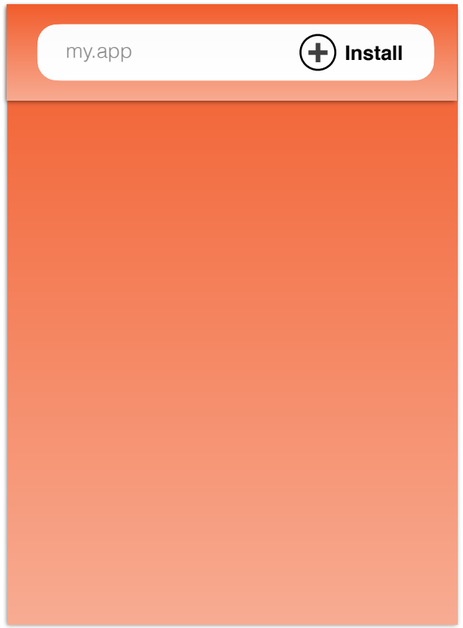
Enterprise

App store

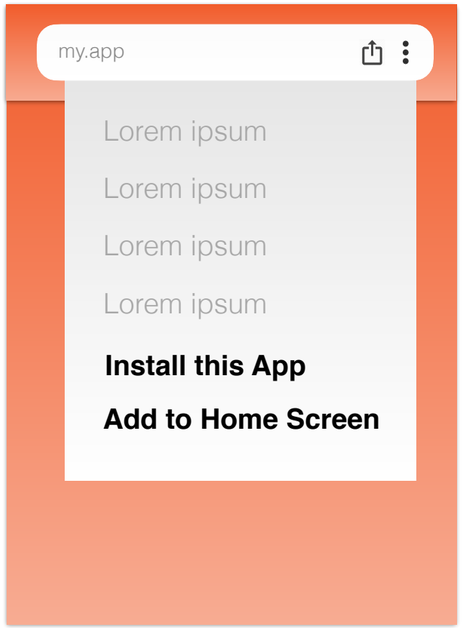
Launcher Icon Installation from Browser



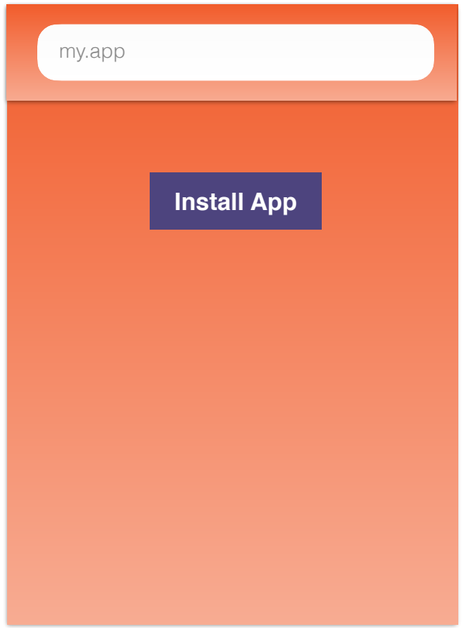
Infobar or banner



Badge



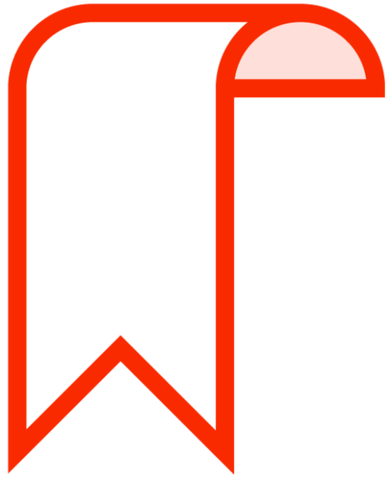
Menu



Custom UI

Launcher Icon Types

Based on installation pattern, browser and host platform

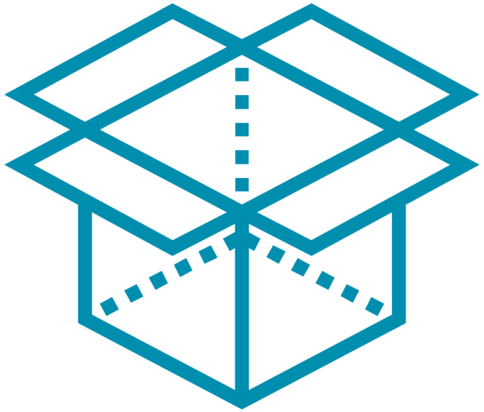


Shortcut or Web Clip

Mobile devices only
It's not a native package

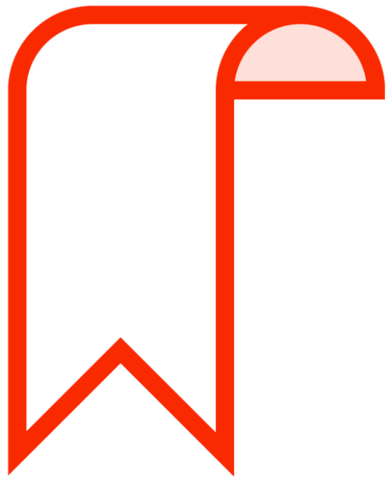
Autogenerated PWA Launcher

Package created and signed by the browser



Manual PWA Launcher

Package created and published by the developer



Shortcut or Web Clip

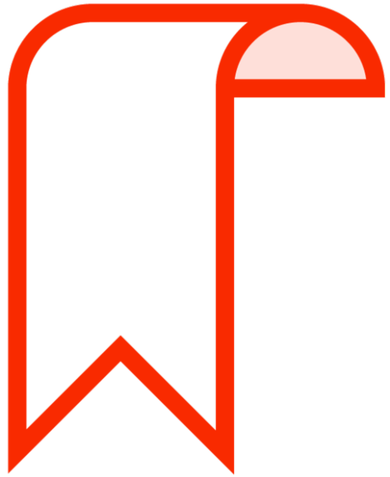
Only available from browser

Opens the browser in a different mode

It's not integrated fully with the host OS

Some limitations:

- No way to detect if it's installed
- Multiple installations are possible
- No URL capture available



Shortcut or Web Clip

iOS and iPadOS:

- Known as **Web Clips**
- They can be installed from Safari or through an Apple Configuration file
- The OS remembers it

Android:

- Known as **Home Screen Shortcuts**
- Firefox, Opera, Brave, Samsung Internet on non-Samsung devices
- Google Chrome on some uncommon situations

Autogenerated PWA Launcher

Only available from browser

It installs a native package

It's created and signed by the browser, locally or in the cloud

Advantages over shortcuts:

- Singleton installations
- URL capture can be available

Some limitations:

- No way to detect if it's installed
- The OS and browser don't remember it

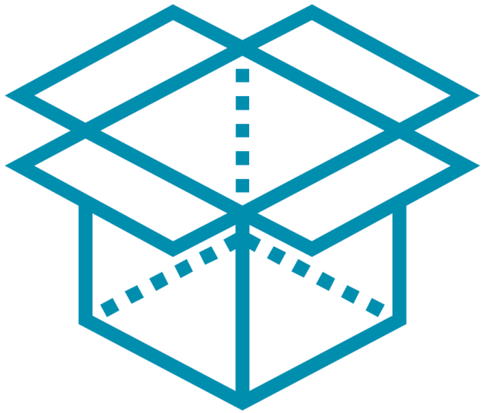
Autogenerated PWA Launcher

Android:

- **WebAPK**, a cloud-generated package
- Google Chrome on any Android device
- Samsung Internet on Samsung Android devices
- Samsung Galaxy Store
- Managed Google Play iframe

Desktop:

- Typically generated locally
- Google Chrome
- Microsoft Edge



Manual PWA Launcher

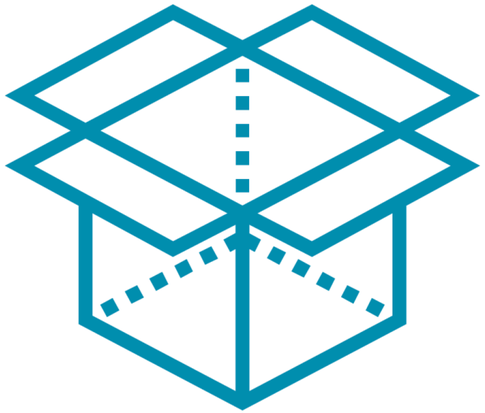
Available from App Stores and Enterprise distribution

Limitations:

- Store rules and requirements

Advantages:

- Singleton installations
- URL capture can be available
- It can be detected on some browsers
- The store and/or OS will remember your installation
- Native APIs available



Manual PWA Launcher

Android:

- Google Play Store
  with TWA (Trusted Web Activity)
- Managed Google Play iframe

iOS and iPadOS:

- App Store with Web View

Windows:

- Microsoft Store

kaiOS:

- kaiOS and JioPhone Stores

What Happens After Installation?

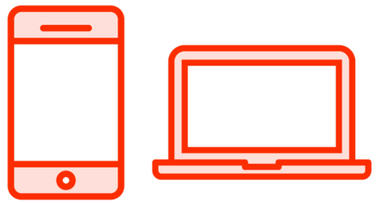
Navigation is transferred to the standalone app

Navigation continues in the browser

App Assets Installation

It downloads and save the necessary resources to render the app on every network situation

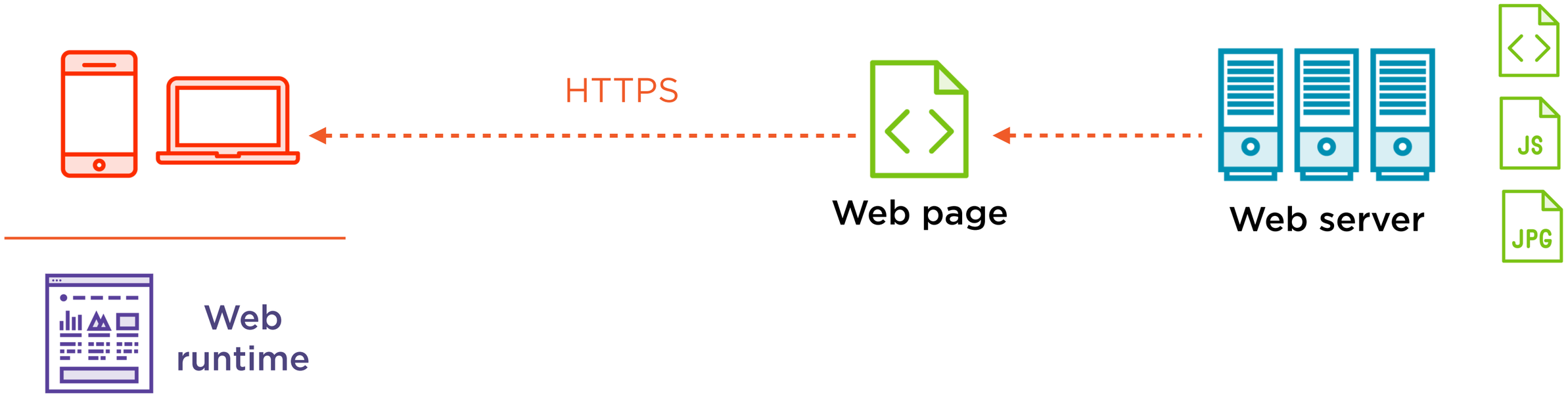
App Assets Installation



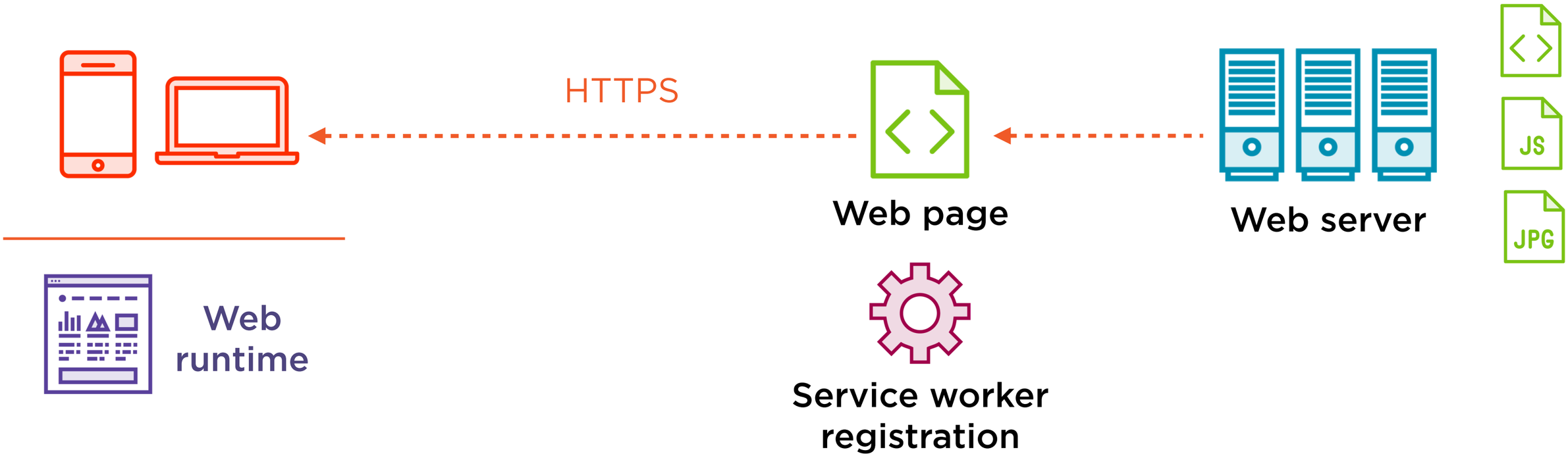
Web runtime

Web server

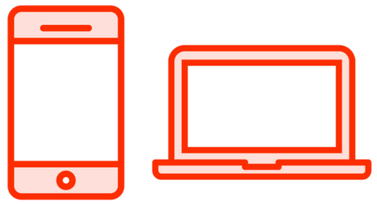
App Assets Installation



App Assets Installation

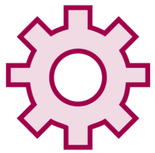


App Assets Installation



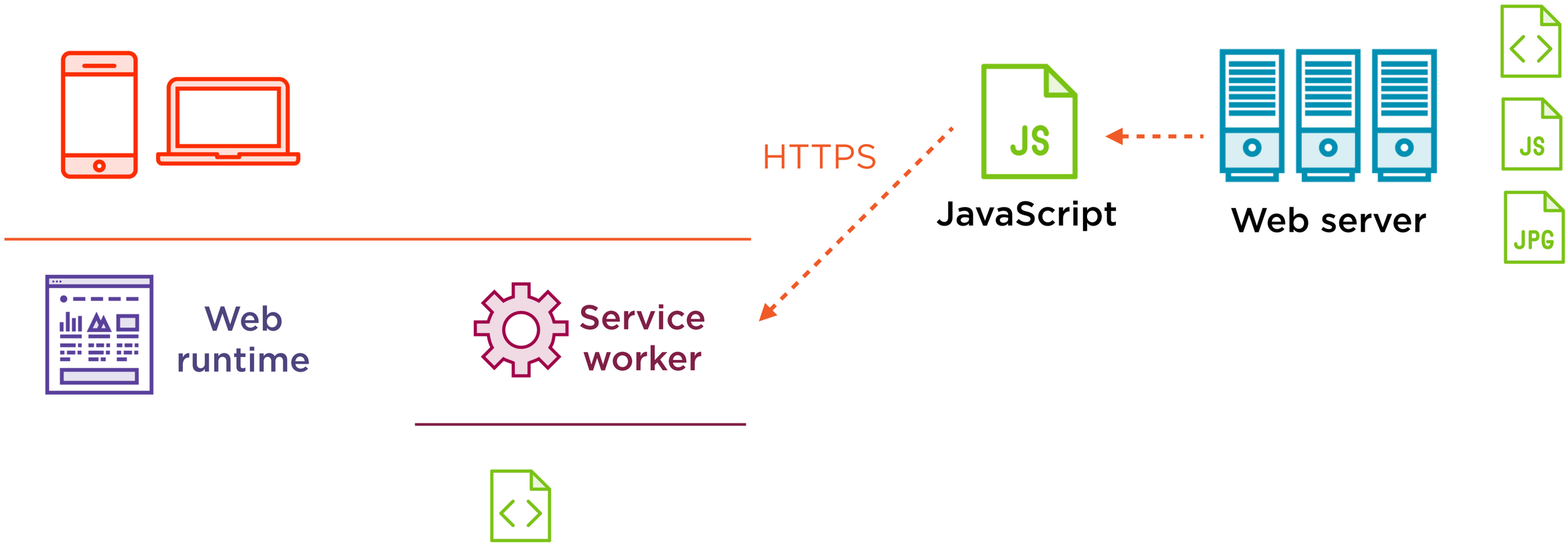
Web server

Web runtime

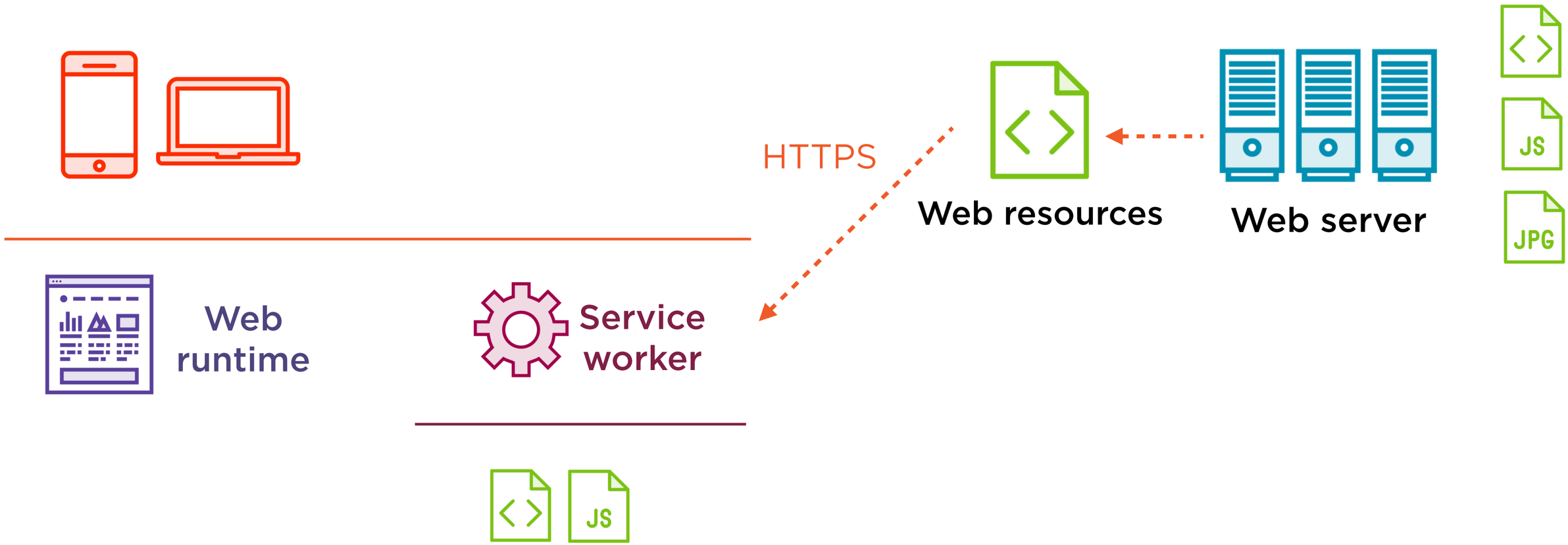


Service worker

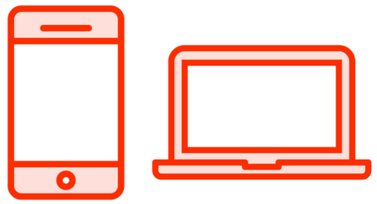
App Assets Installation



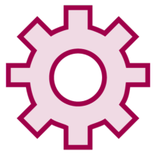
App Assets Installation



App Assets Installation

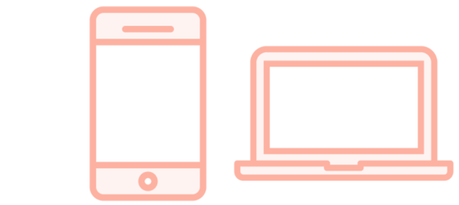


Web runtime

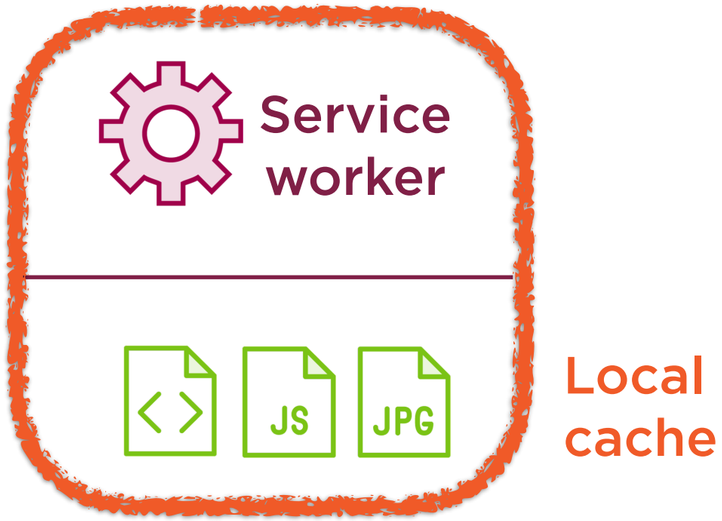


Service worker

App Assets Installation



Web runtime



Local cache

When does the App Assets Installation Happen?

Using the PWA from the browser

Using the PWA for the first time after installation
(iOS & iPadOS only)

Using the PWA for the first time after installing it from the app store

App Assets and Shortcut Icon installations happen independently

Partial Installations

Launcher Icon but
no App Assets

App Assets but
no Launcher Icon

Summary

Installation Architecture

- Icon installation and App Assets installation
- Many types of icon installation: browser, autogenerated launcher, manual launcher
- App Assets with Service Worker

Track PWA Installations for Analytics

Measuring Impact for Installation

Installation availability

Promotional info-bar presentation

Launcher icon install rates

App assets install rates

Not every metric available on every platform

Analytics we can not track

Uninstallation

Track metrics

You can use your own custom code or SDKs such as Google Analytics

script.js

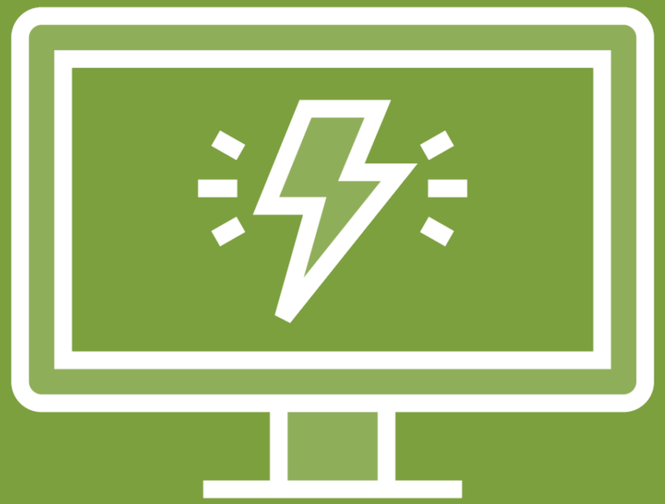
```
// Google Analytics SDK
function track(action, value) {
    ga('send', 'event', action, value);
}
```

Track metrics

You can use your own custom code or SDKs such as Google Analytics

script.js

```
// Google Analytics SDK
function track(action, value) {
    ga('send', 'event', action, value);
}

// Tracking installation availability
track('install', 'available');
// Tracking PWA installation
track(install', 'installed');
```



Tracking Launcher Icon Installation on Chromium

Google Chrome, Opera, Microsoft Edge, Brave and other browsers.

Detect installation availability

Available only on Chromium-based browser

script.js

```
// Tracking installation availability
window.addEventListener('beforeinstallprompt', event => {
  track('install', 'available');
});
```

beforeinstallprompt Event

It was part of Web App Manifest spec

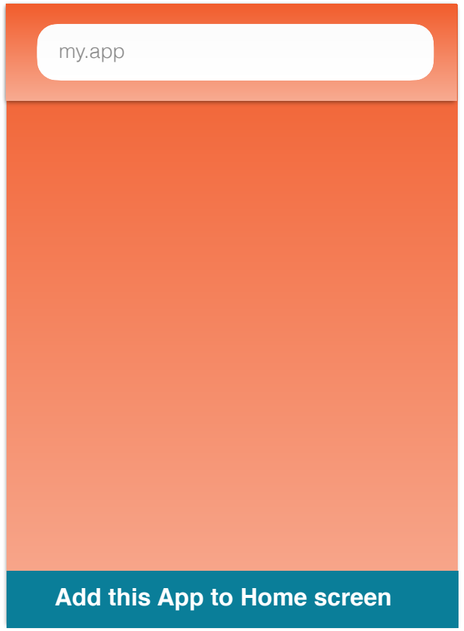
Now implemented by Chromium only

Event is fired when:

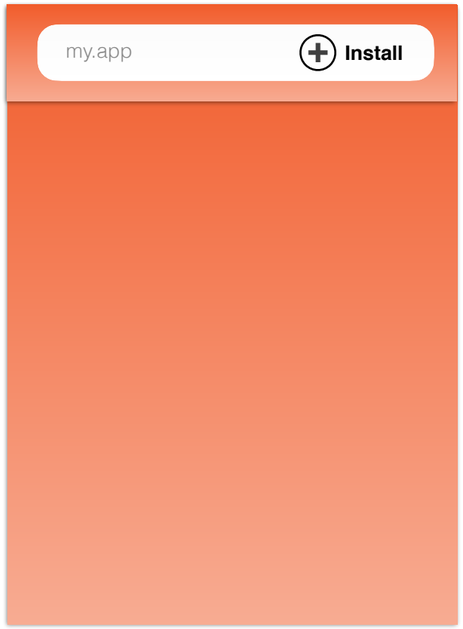
- Current URL passes the PWA criteria
- The PWA is not already installed

Installation Availability Will...

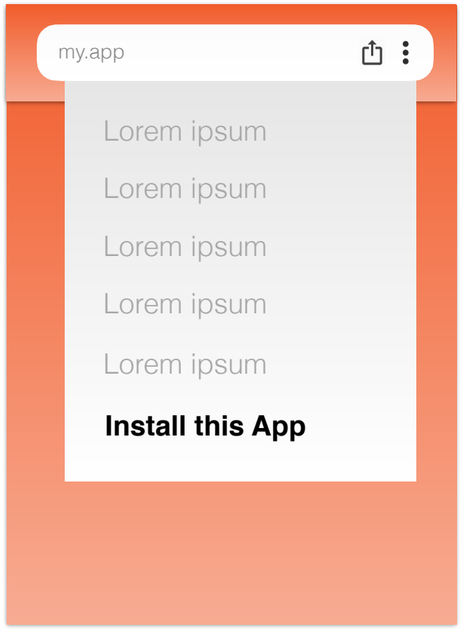
fire ⚡beforeinstallprompt event



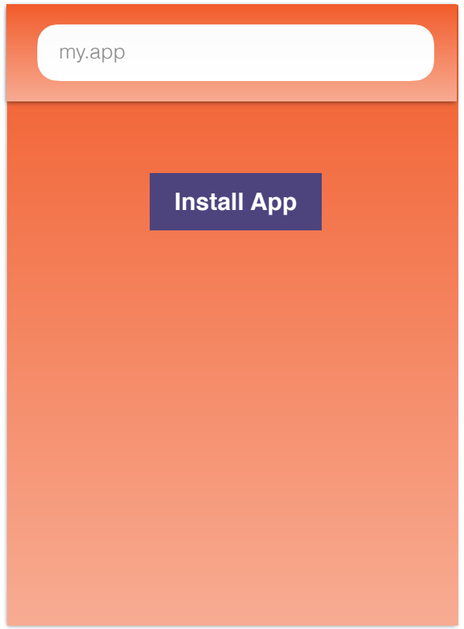
Render infobar or banner
(cancellable)



Add a badge



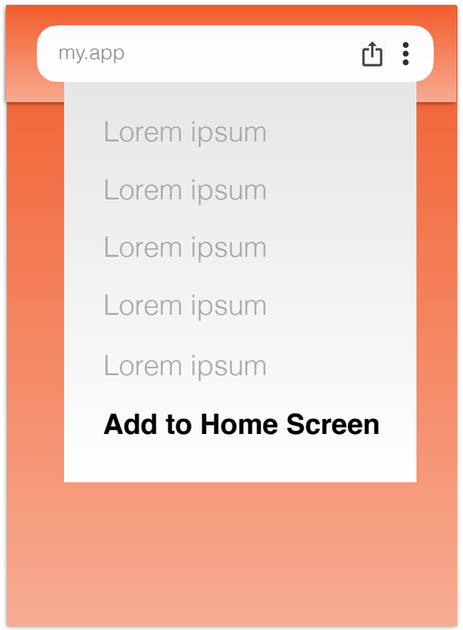
Create a desktop menu item



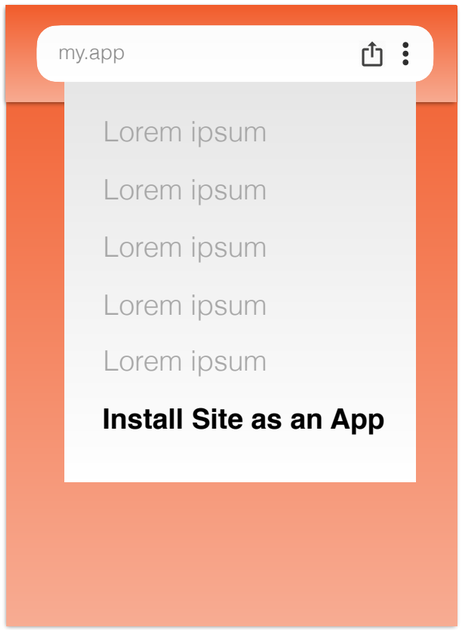
Allow developer to provide a custom install button

There are always-available installation menu items that can't be tracked

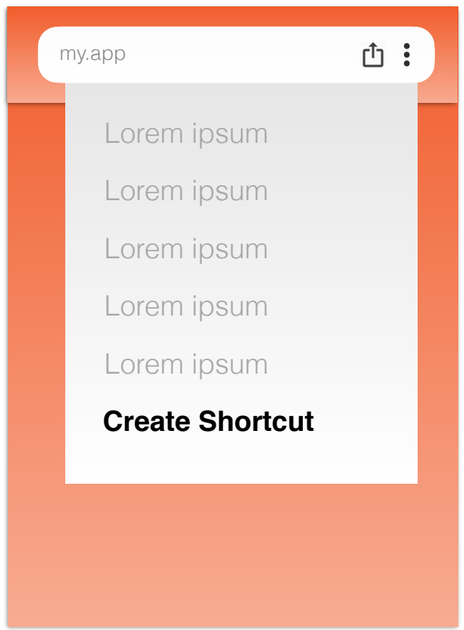
Always-available Installation Menu Items



Browsers on Android, iOS and iPadOS



Microsoft Edge on Desktop



Chrome on Desktop

Detect installation

Available only on Chromium-based browser

script.js

```
// Tracking PWA installation on Chromium
window.addEventListener('appinstalled', event => {
  track('install', 'installed');
});
```

appinstalled Event

It was part of Web App Manifest spec

Now implemented by Chromium only

Fired when:

- The user has accepted the native installation dialog from menu item, badge, infobar or custom UI element

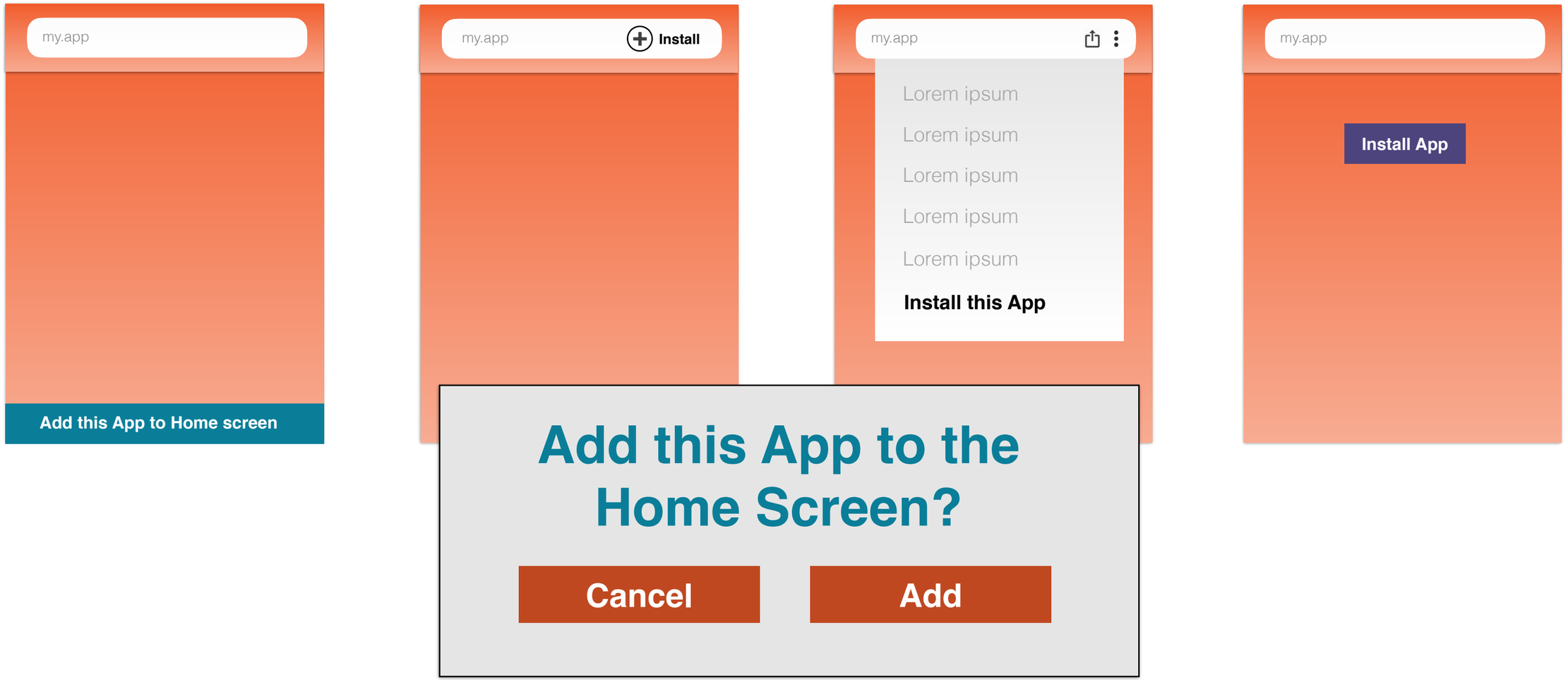
Warnings:

- On Android with WebAPK, the package generation takes some seconds

- On Android 8+ with Shortcuts, the user will have to pick where to save it and she can cancel as well after

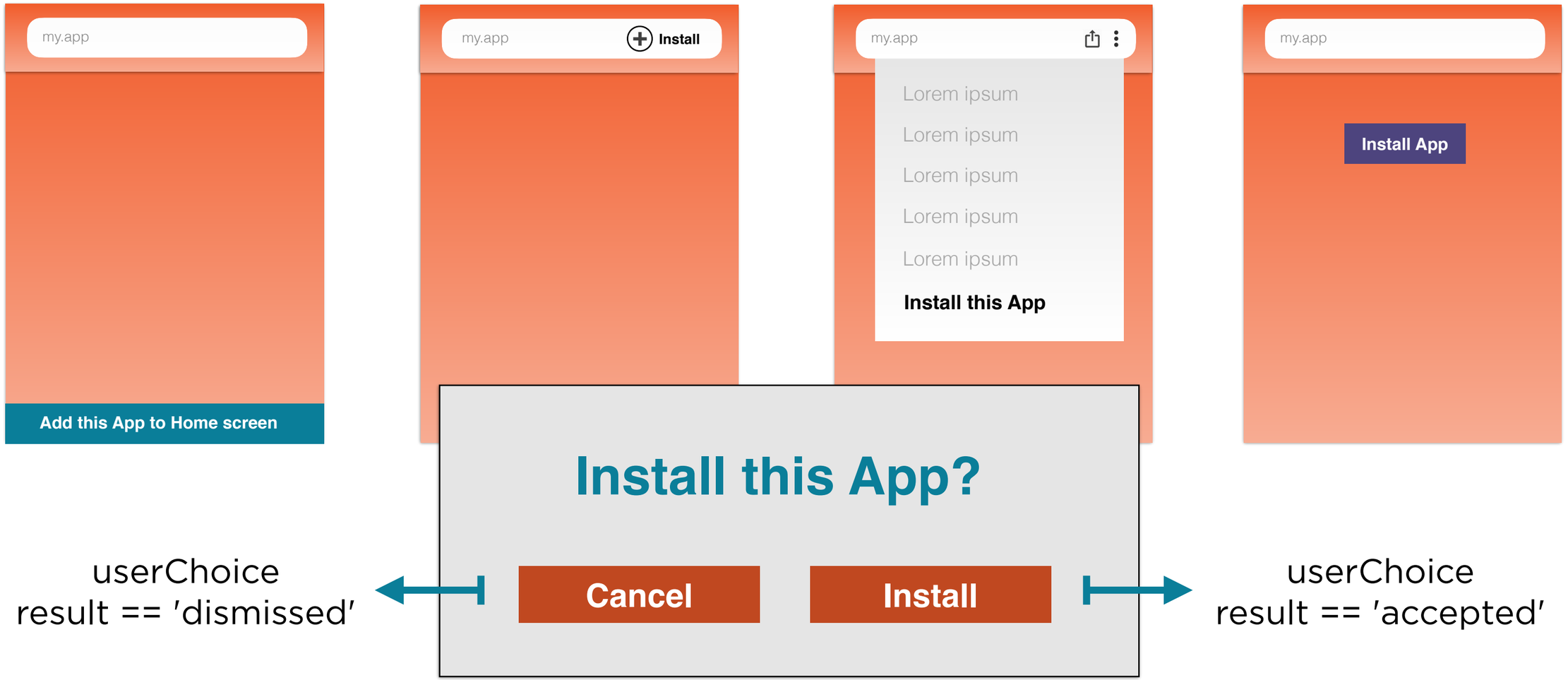
Chromium Installation Flow

⚡beforeinstallprompt event fired

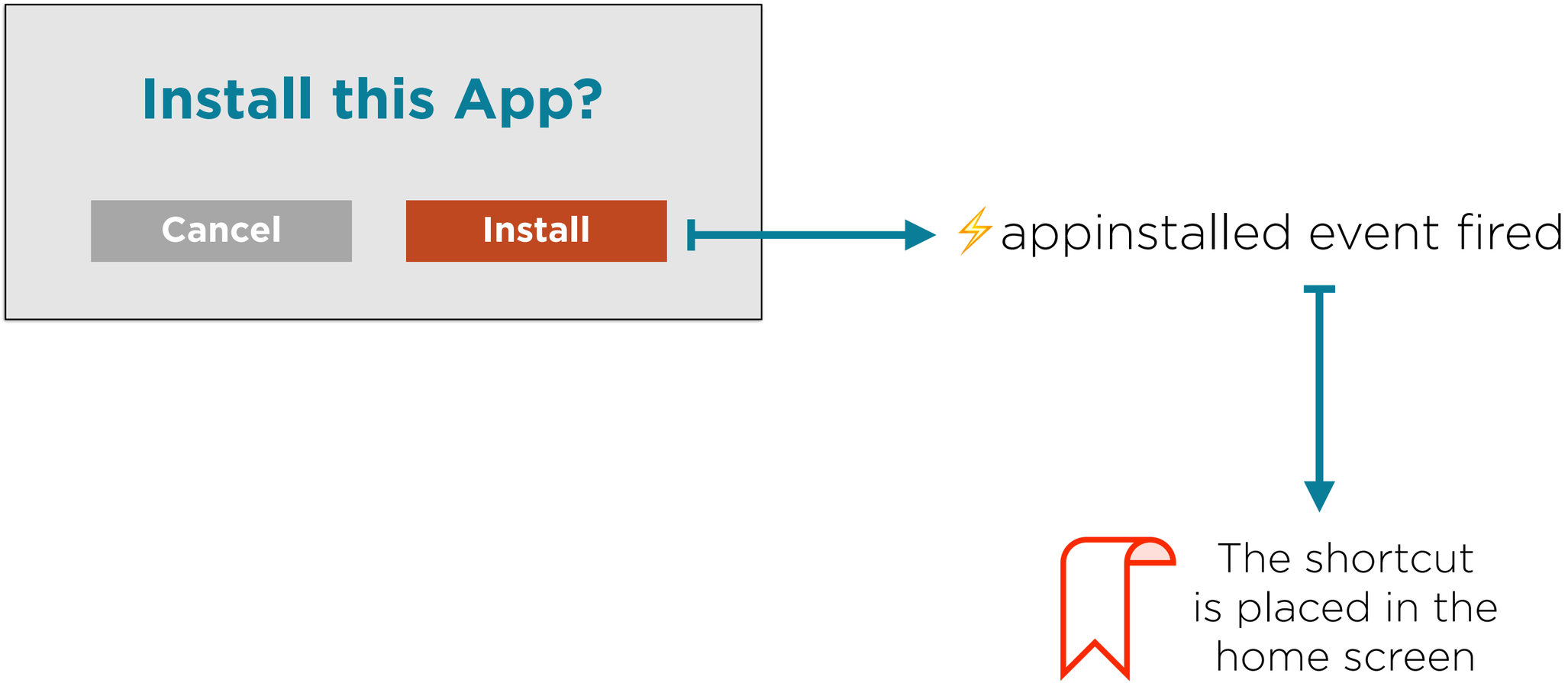


Chromium Installation Flow

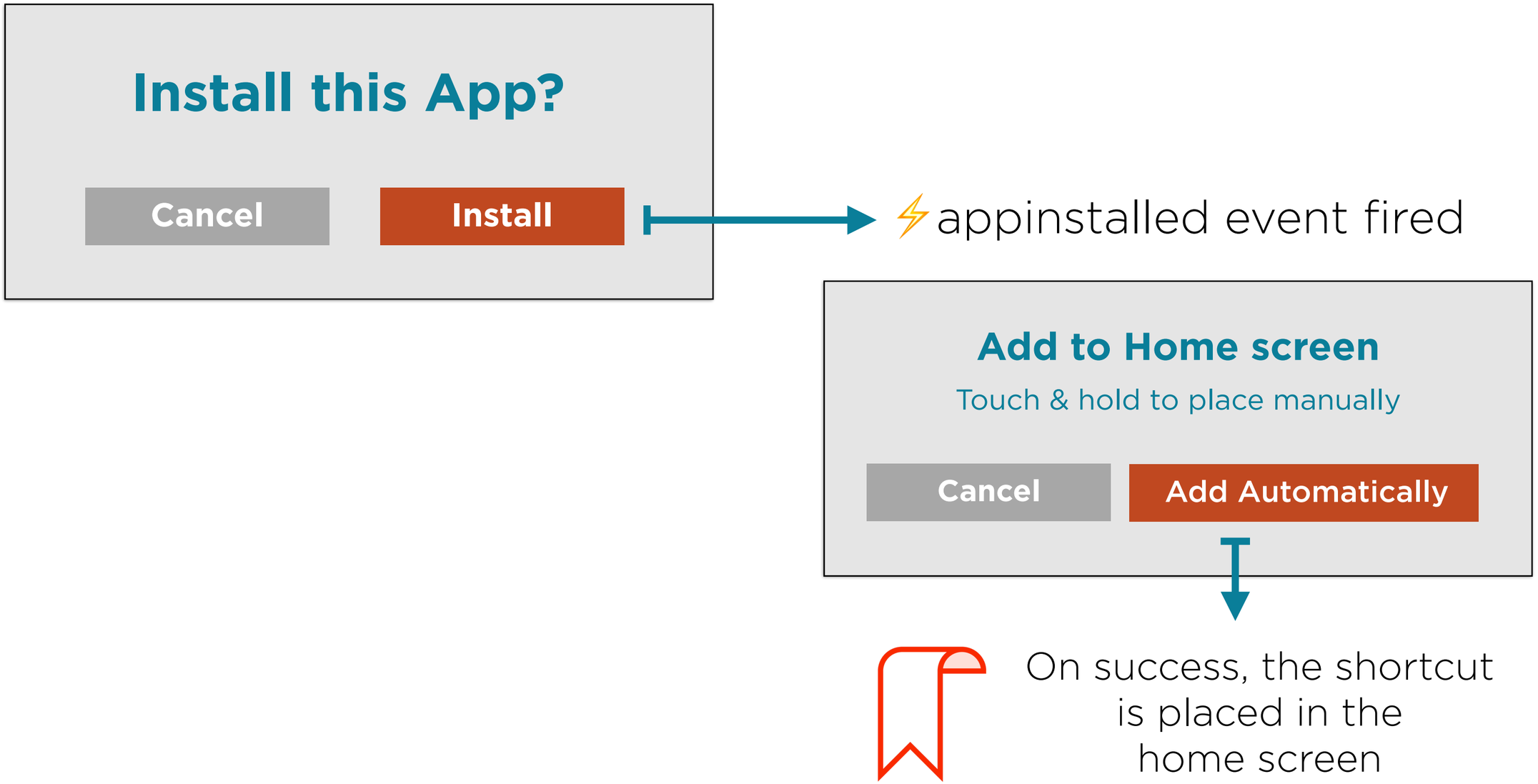
⚡beforeinstallprompt event fired



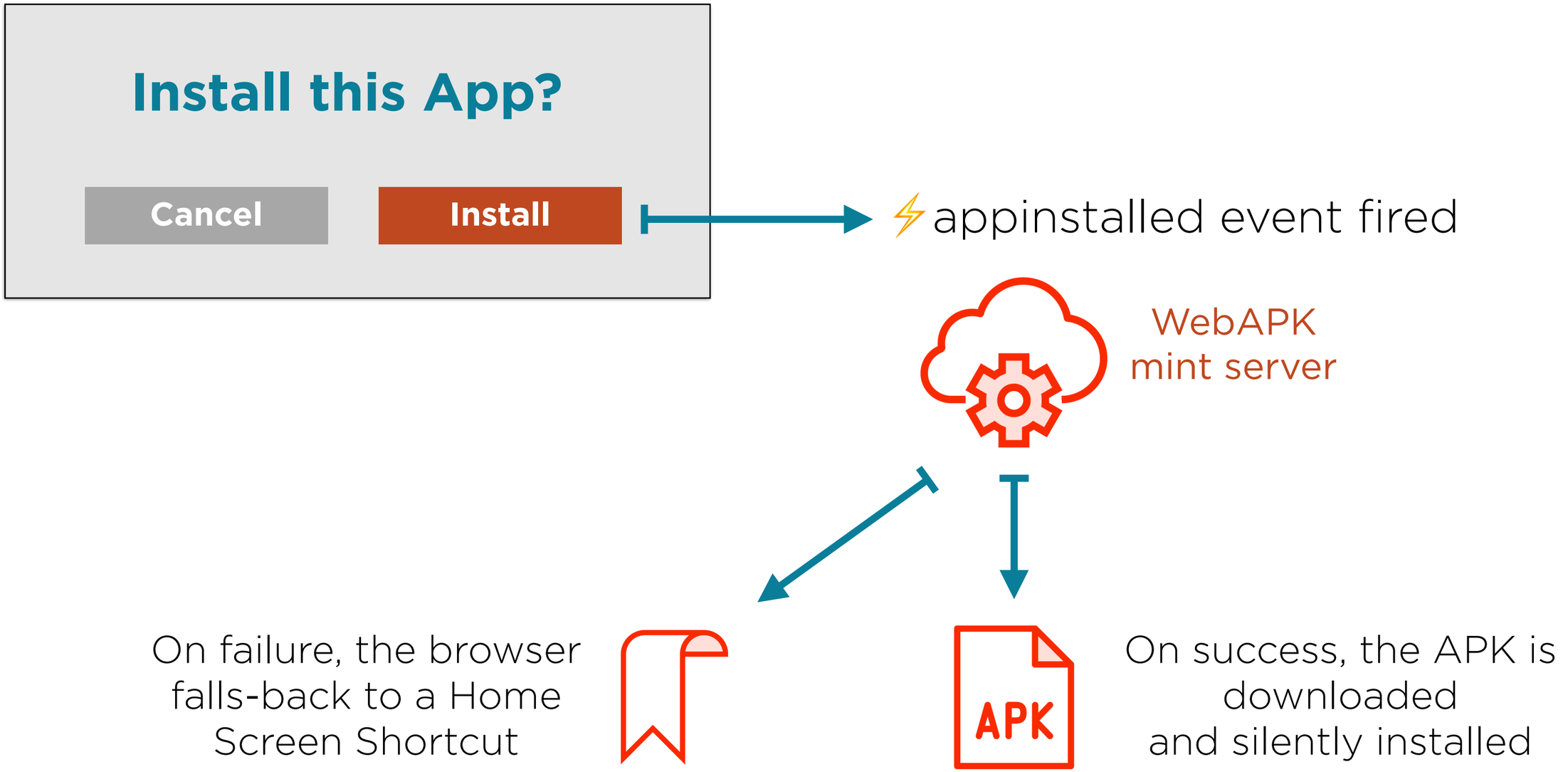
Chromium Installation Flow: Android 4-7 Shortcuts



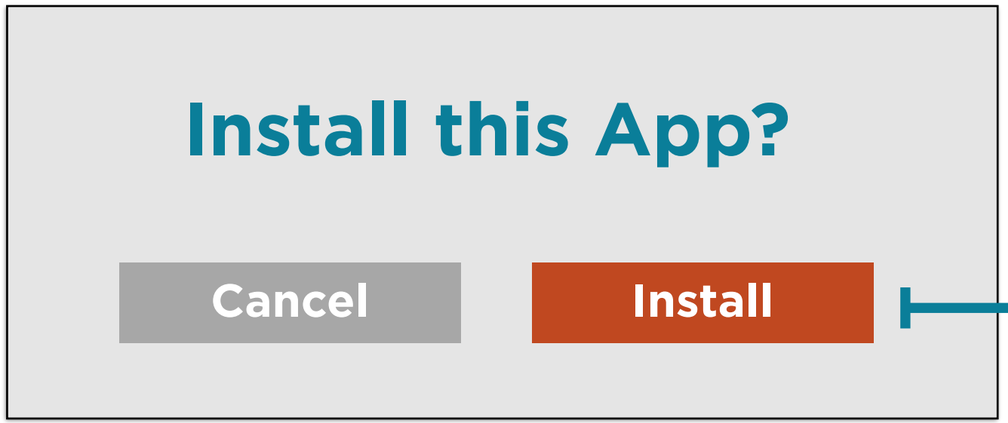
Chromium Installation Flow: Android 8+ Shortcuts



Chromium Installation Flow: Android WebAPK



Chromium Installation Flow: Desktop



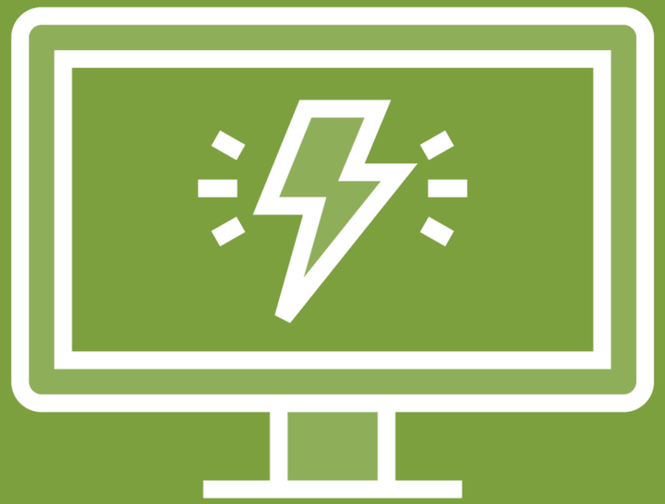
appinstalled event fired

The PWA launcher is generated and installed

Browser's tab is closed; navigation continues in the PWA scope

All these tracking methods are not available on Safari or Firefox

If users are installing the app through an app store, you should look at analytics provided by them



Tracking Launcher Icon Installation on Safari

While it's not possible to detect it, we can make a fair guess using some tricks

On iOS and iPadOS a PWA launcher icon has a different storage than Safari for the same origin

Detect installation on iOS and iPadOS

When the PWA is installed an empty sandbox is used for storage

script.js

```
// Tracking deferred installation on iOS and iPadOS
window.addEventListener('DOMContentLoaded', event => {
  if (navigator.standalone)




  }
});
```

Detect installation on iOS and iPadOS

When the PWA is installed an empty sandbox is used for storage

script.js

```
// Tracking deferred installation on iOS and iPadOS
window.addEventListener('DOMContentLoaded', event => {
  if (navigator.standalone &&
        localStorage.getItem('firstLoad')===undefined) {




  }
});
```

Detect installation on iOS and iPadOS

When the PWA is installed an empty sandbox is used for storage

script.js

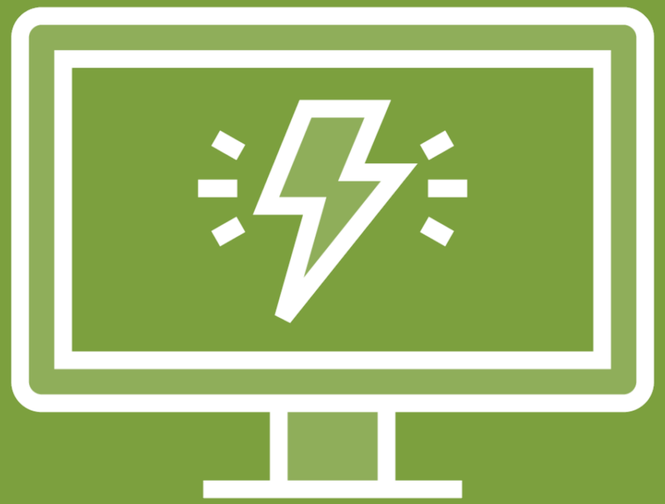
```
// Tracking deferred installation on iOS and iPadOS
window.addEventListener('DOMContentLoaded', event => {
  if (navigator.standalone &&
        localStorage.getItem('firstLoad')===undefined) {
    track('install', 'installed');



  }
});
```

Detect installation on iOS and iPadOS

When the PWA is installed an empty sandbox is used for storage

script.js

```
// Tracking deferred installation on iOS and iPadOS
window.addEventListener('DOMContentLoaded', event => {
  if (navigator.standalone &&
      localStorage.getItem('firstLoad')===undefined) {
    track('install', 'installed');
    // Set a flag for future loads on this PWA icon instance
    // WARNING: false positives can happen
    localStorage.setItem('firstLoad', false);
  }
});
```



Tracking App Assets Installation

Using the Cache Storage API typically from a Service Worker we can track when all the assets are available client-side. This is multi-vendor.

App Assets Installation Tracking

Available on all browsers

serviceworker.js

```
// Tracking app asset installation
self.addEventListener('install', event => {









});
```

App Asset Installation Tracking

Available on all browsers

serviceworker.js

```
// Tracking app asset installation
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('appCache')




  );
});
```

App Asset Installation Tracking

Available on all browsers

serviceworker.js

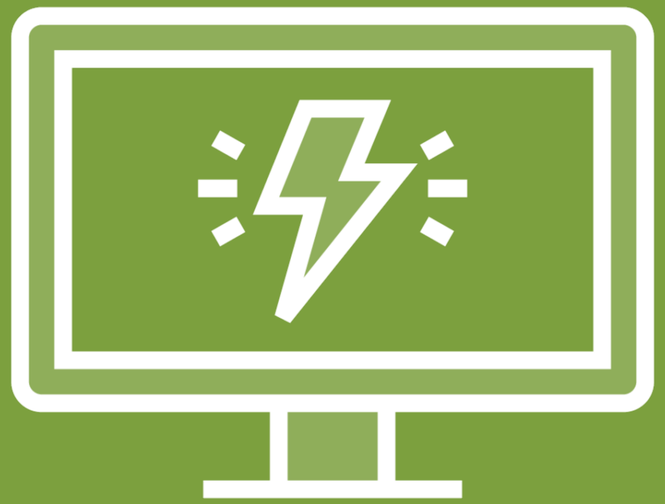
```
// Tracking app asset installation
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('appCache')
          .then(cache => cache.addAll(listOfAssets)


        );
  );
});
```

App Asset Installation Tracking

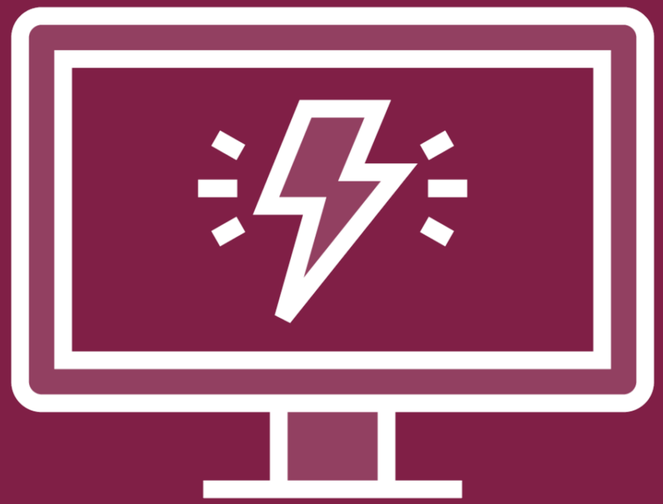
Available on all browsers

serviceworker.js

```
// Tracking app asset installation
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('appCache')
          .then(cache => cache.addAll(listOfAssets)
                              .then(event => track('assets', 'installed'))
                              .catch(event => track('assets', 'failed'))
                )
  );
});
```



Tracking Uninstallation



Tracking Uninstallation

It's not possible to detect when the user is uninstalling the App or if the cache is cleared. Only PWAs in the Store can track uninstalls.

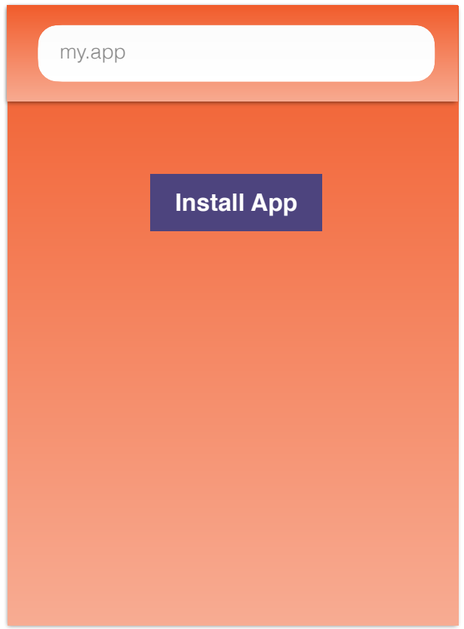
Summary

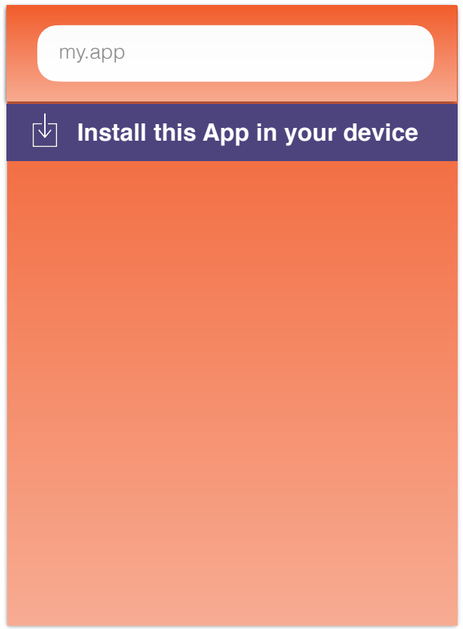
Tracking Installation

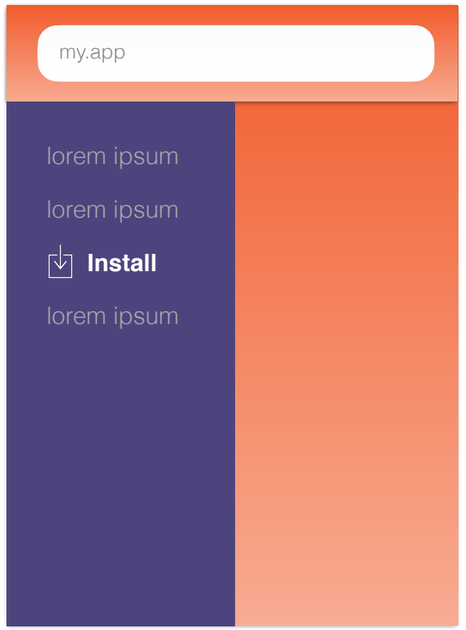
- Availability: **beforeinstallprompt** event
- Icon Installation: **appinstalled** event
- Assets Installation: service worker events
- False positives can occur on appinstalled
- Most events Chromium-only
- Tricks for iOS and iPadOS
- No uninstallation tracking available

Promote Installation from the User Interface

Create a Custom Install Promotion







Custom Install Promotion

It's based on beforeinstallprompt event

Chromium-based browsers only

We need to wait for the event

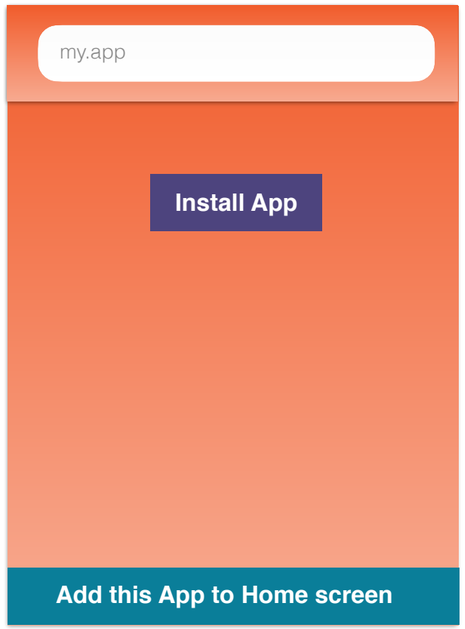
Hide the promotion if:

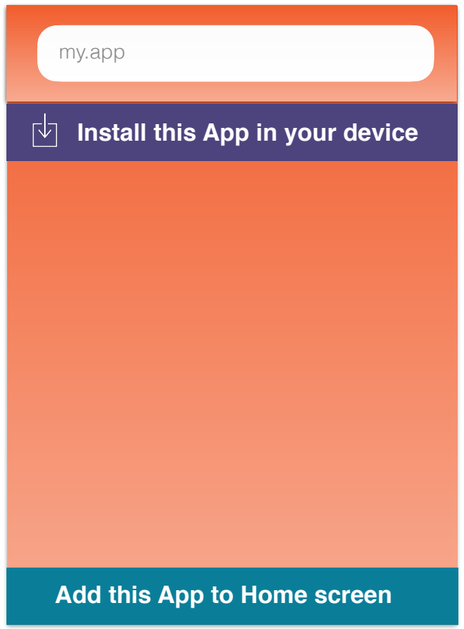
- The user is not in browser display mode
- You know the app is installed
- You know an alternative app is installed

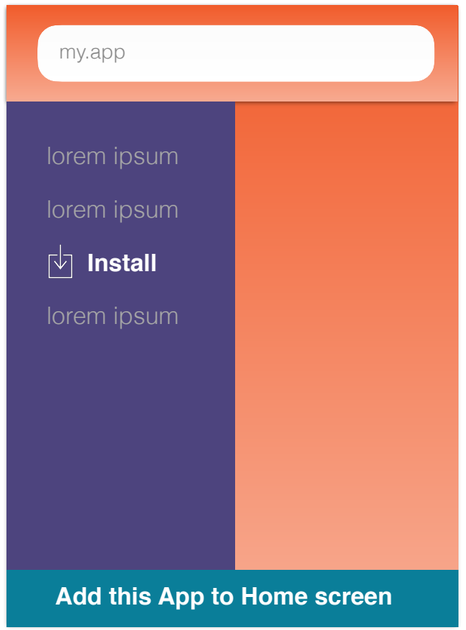
While no event is fired you can:

1) Hide the promotion
2) Fallback to manual instructions

Avoid user experience duplications







Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {



});
```

Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  // we prevent a banner or infobar to appear
  event.preventDefault();


});
```

Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
// we save the prompt event in a global variable
let installPromptEvent;

window.addEventListener('beforeinstallprompt', event => {
  // we prevent a banner or infobar to appear
  event.preventDefault();


});
```

Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
// we save the prompt event in a global variable
let installPromptEvent;

window.addEventListener('beforeinstallprompt', event => {
  // we prevent a banner or infobar to appear
  event.preventDefault();
  // we save the event for later usage
  installPromptEvent = event
});
```

Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
// we save the prompt event in a global variable
let installPromptEvent;

installButton.addEventListener('click', event => {




});
```

Promote Install through a Custom UI

Available only on Chromium-based browser

script.js

```
// we save the prompt event in a global variable
let installPromptEvent;

installButton.addEventListener('click', event => {
    if (installPromptEvent) {
        installPromptEvent.prompt();
    } else {
        // Prompt not available: Display fallback instructions
    }
});
```

Detect installation prompt result

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  ...
  // Tracking installation prompt result




});
```

Detect installation prompt result

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  ...
  // Tracking installation prompt result
  event



});
```

Detect installation prompt result

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  ...
  // Tracking installation prompt result
  event.userChoice



});
```

Detect installation prompt result

Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  ...
  // Tracking installation prompt result
  event.userChoice.then(result => {


  });
});
```

Detect installation prompt result

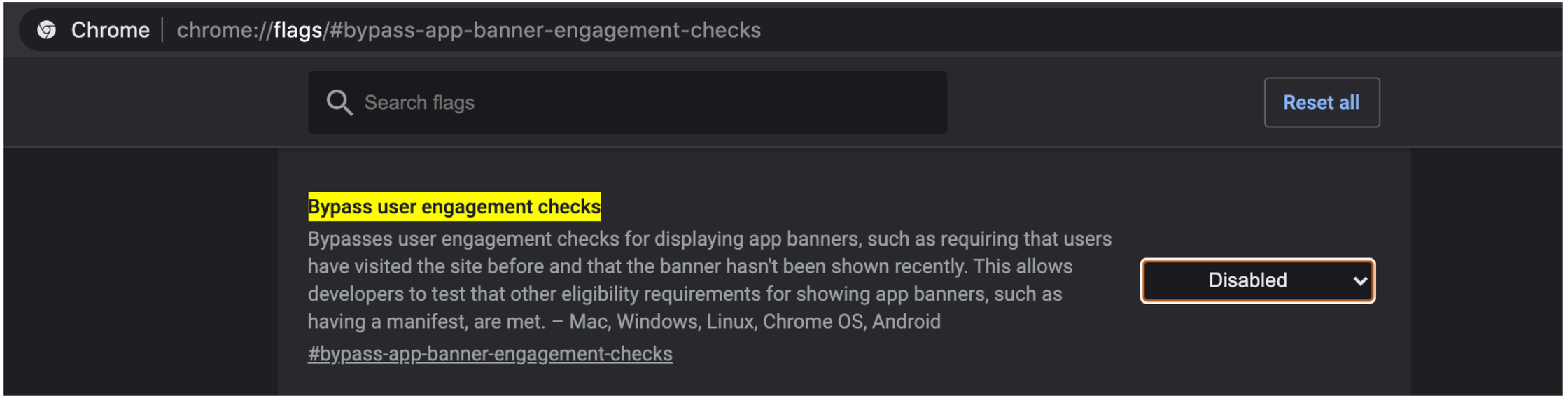
Available only on Chromium-based browser

script.js

```
window.addEventListener('beforeinstallprompt', event => {
  ...
  // Tracking installation prompt result
  event.userChoice.then(result => {
    // result.outcome can be 'accepted' or 'dismissed'
    track('installprompt', result.outcome);
  });
});
```

If the app is already installed on desktop or Android with WebAPK, event won't be fired

Chromium Flag to Better Debugging



Prioritize Store App over Browser Installation

App in the Store over the Browser's PWA

Native App
(Not a PWA)

PWA using Trusted Web Activity on Play Store

PWA using Web View on App Store or Microsoft Store

Store Apps vs. Browser PWAs

Store app:

- can be remembered in the account
- may use more features
- have more abilities granted

Avoid app duplication:

- If the store app is installed, it's better to avoid install promotion from the browser
- If the store app is not installed, we can select which app to promote

Technologies Available

Installed Related Apps API

Web App Manifest Related App

Apple's Smart App Banner

We need to know the native app's package ID, typically in the form of:

com.domain.app-name

These techniques don't work with WebAPK on Android. They have a package ID but it's internal and not available for the developer

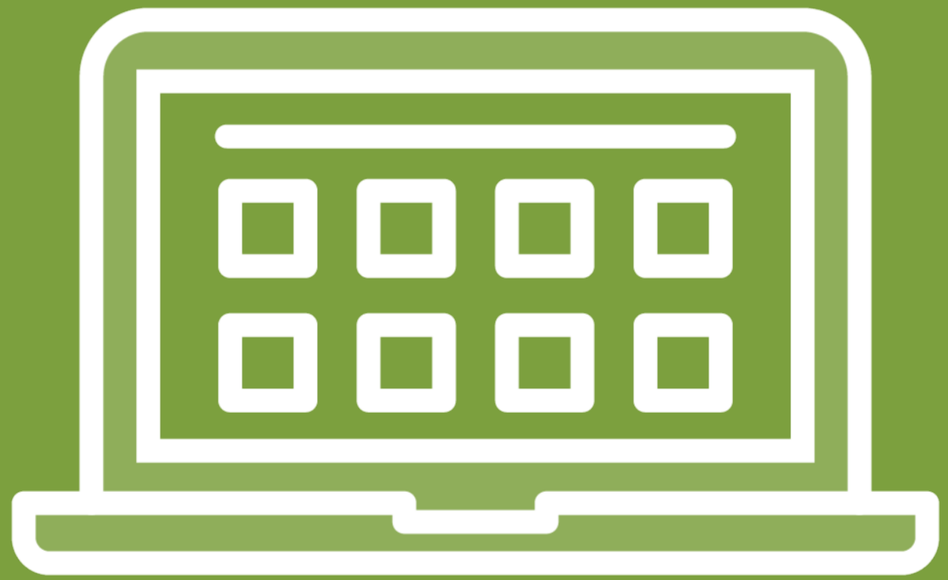
Related Apps in App Manifest

The Web App Manifest let us define related store-based apps to the current website

```
{
  "name": "PWA Name",
  "display": "standalone",
  "scope": "/",
  "start_url": "/",
  "related_applications": [
    {
     "platform": "play",
     "id": "com.myapp.pwa",
     "url": "https://
play.google.com/store/apps/
details?id=com.myapp.pwa",
    }
  ]
}
```

Web app manifest

- **Related Apps Collection**
  - Platform:
    "**play**" is for Google Play Store
    "**itunes**" is for App Store
    "**windows**" is for Microsoft Store
  - URL
  - Id is the Package Id



Installed Related Apps API

Available on some Chromium-based browsers (Edge, Chrome) at least on Android and Windows

Get Installed Related Apps

Available only on Chromium-based browsers - Not a standard yet

script.js

```
// we get all the installed related apps taken from Manifest
const installedApps = await navigator.getInstalledRelatedApps();
```

Get Installed Related Apps

Available only on Chromium-based browsers - Not a standard yet

script.js

```
// we get all the installed related apps taken from Manifest
const installedApps = await navigator.getInstalledRelatedApps();
// we filter them to find the one we need by package ID
const packageId = "com.app.pwa";
const app = installedApps.find(app => app.id === packageId);
```

Get Installed Related Apps

Available only on Chromium-based browsers - Not a standard yet

script.js

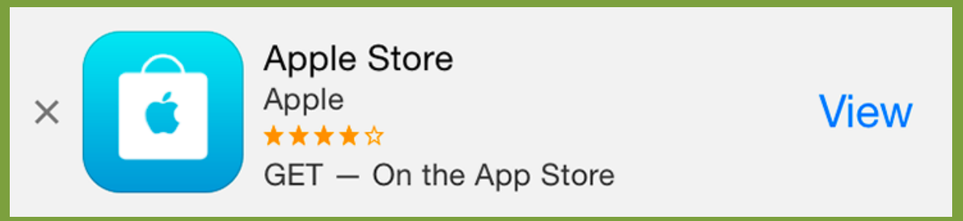
```
// we get all the installed related apps taken from Manifest
const installedApps = await navigator.getInstalledRelatedApps();
// we filter them to find the one we need by package ID
const packageId = "com.app.pwa";
const app = installedApps.find(app => app.id === packageId);
if (app) {
    // app was found
    console.log(`${app.id} version ${app.version} is installed`);
}
```

If the app and version we want is already installed we should not promote PWA installation from browser

```
{
  "name": "PWA Name",
  "display": "standalone",
  "scope": "/",
  "start_url": "/",
  "prefer_related_applications":
     true,
  "related_applications": [
    {
     "platform": "play",
     "id": "com.myapp.pwa",
     "url": "https://
play.google.com/store/apps/
details?id=com.myapp.pwa",
    }
  ]
}
```

Web app manifest

- **Prefer Related Apps Flag**
  - False by default
  - It express to the browser if we prefer the store app against PWA installed from the browser



Smart App Banner

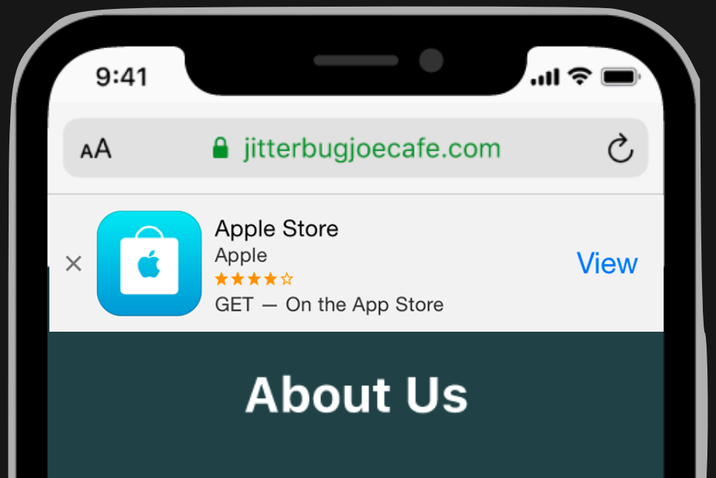
Available only on Safari on iOS and iPadOS; it connects a website to a corresponding app in the AppStore, including App Clips. No API is available to use it.

Smart App Banners

Meta tag useful only for iOS and iPadOS

index.html

```
<meta name="apple-itunes-app"
content="app-id=com.myapp.pwa">
```



Smart App Banners

Meta tag useful only for iOS and iPadOS

index.html

```
<meta name="apple-itunes-app"
content="app-id=com.myapp.pwa,app-argument=myapp.com/deep/link">
```

Smart App Banners for App Clips

Small store app that will run without installation from Safari

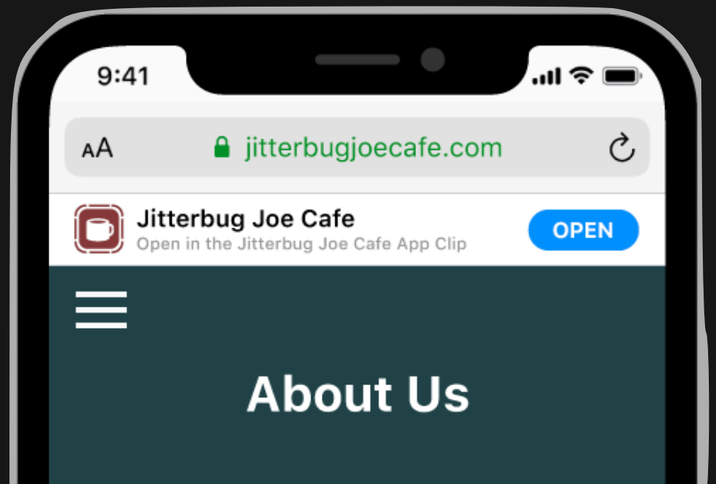
index.html

Smart App Banners for App Clips

Small store app that will run without installation from Safari

index.html

```
<meta name="apple-itunes-app"
content="app-clip-bundle-id=com.myapp.pwa.clip">
```



Summary

If you have apps in the Store (PWA or not)

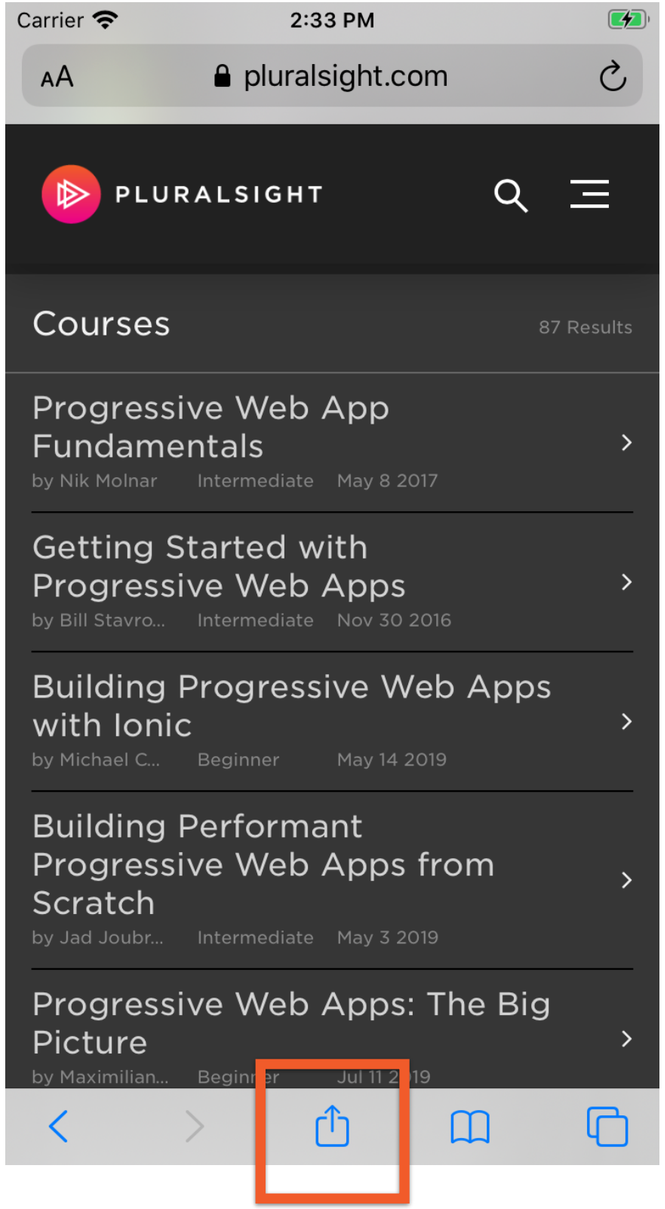
- Define Related Apps in Web App Manifest
- Define your installation preference
- Use Smart App Banners for iOS and iPadOS
- Use Installed Related Apps API to decide internal promotions in your user interface

Improve Installation Reliability on iOS and iPadOS

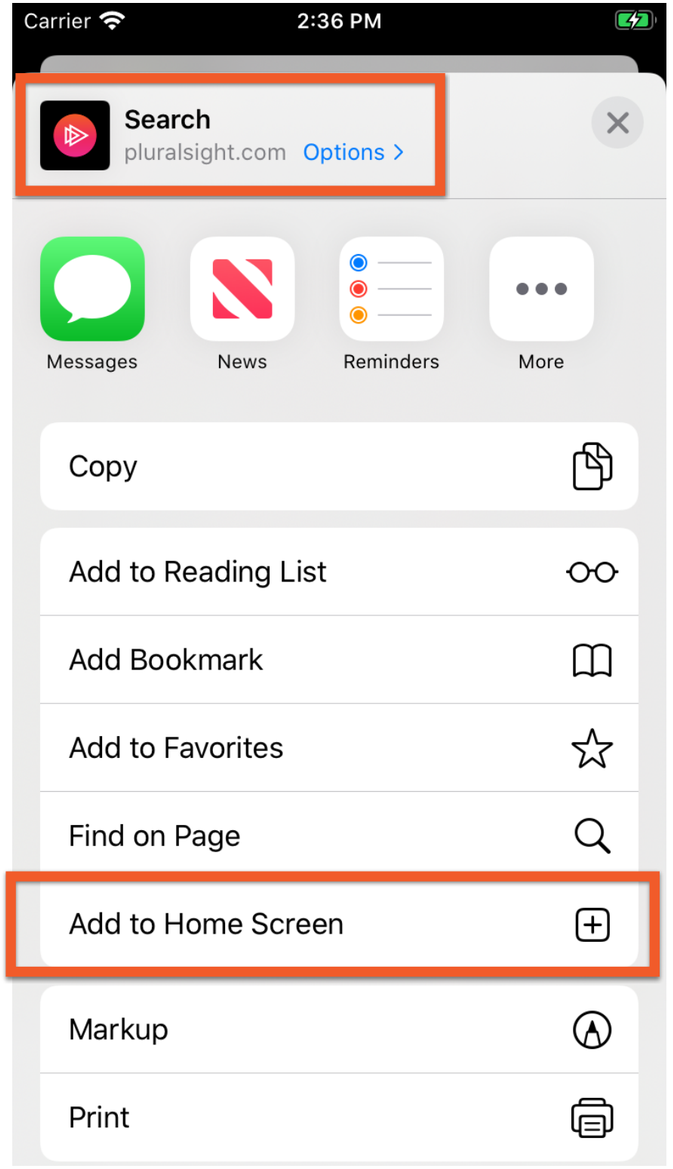
Loading the App Manifest

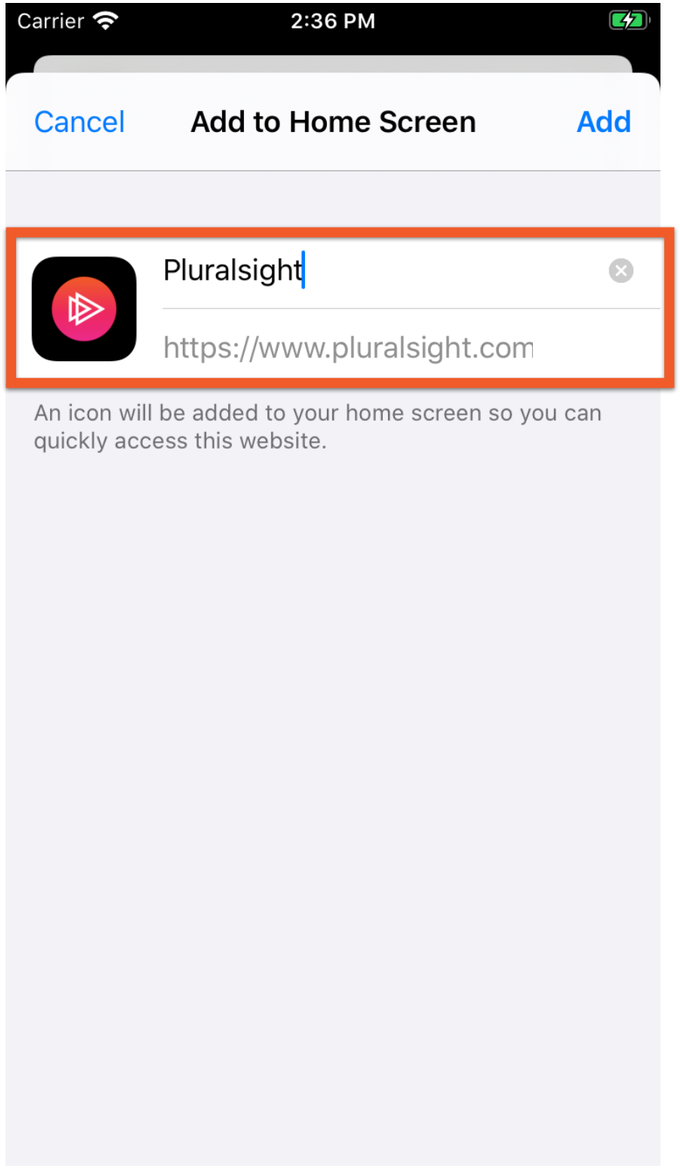
Most browsers download and parse the App Manifest when the PWA loads in the tab

But Safari loads the App Manifest when the user opens Share menu



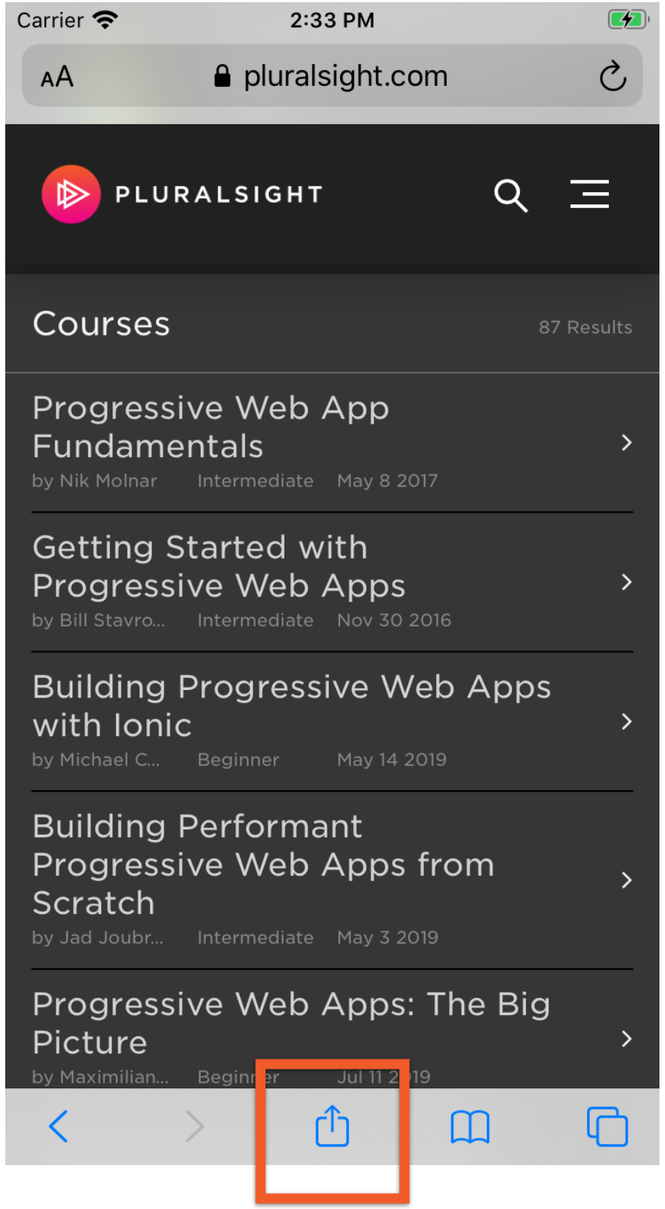
Safari loads App Manifest and icon



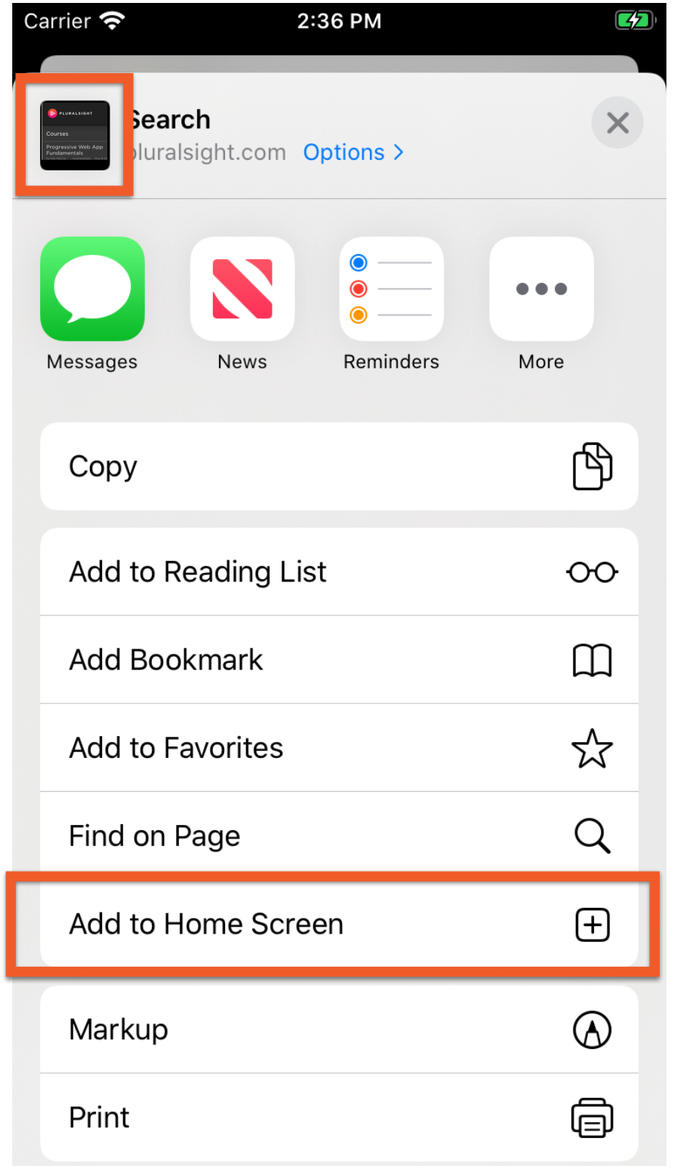


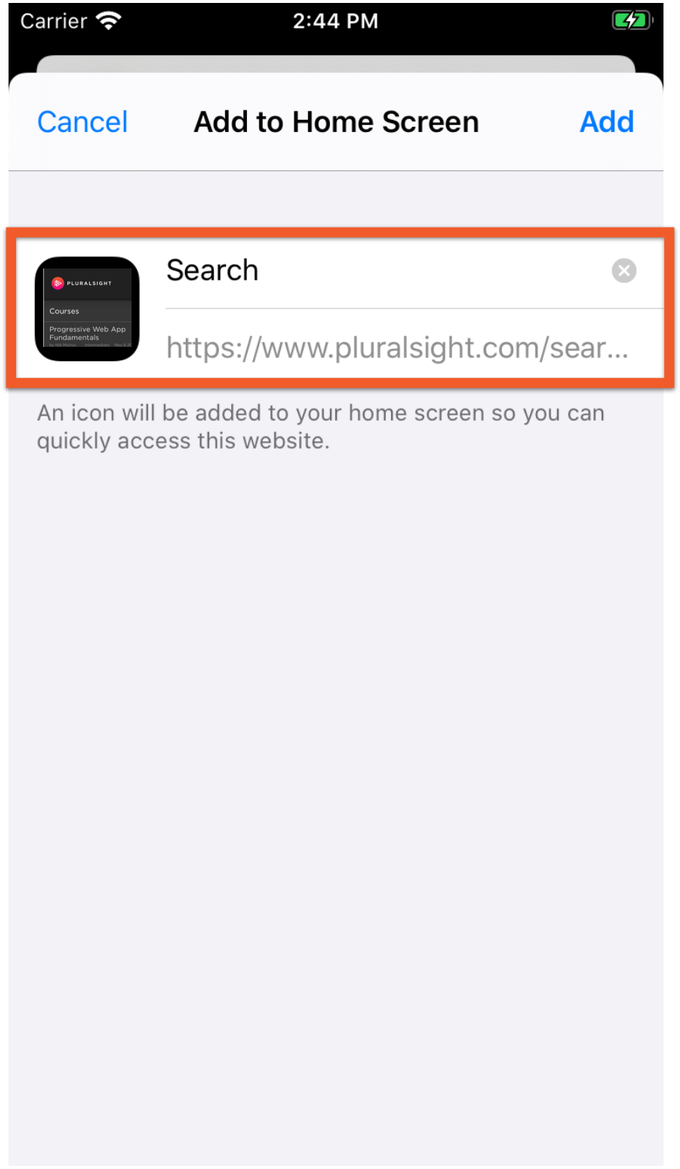
App Manifest Not Loading

Sometimes the user installs the PWA but the App Manifest couldn't be loaded



Safari can't load manifest or icon





If there is a network issue or the manifest takes more than a couple of seconds to load, Safari will fallback to a browser shortcut with no manifest

Icon will be a screenshot and the link will point to current URL and not start_url

Preload App Manifest and icons for Safari

index.html

```
<link rel="manifest" href="app.webmanifest">
<link rel="preload" href="app.webmanifest" as="manifest">
```

Preload App Manifest and icons for Safari

index.html

```
<link rel="manifest" href="app.webmanifest">
<link rel="preload" href="app.webmanifest" as="manifest">

<link rel="apple-touch-icon" href="apple-icon.png">
<link rel="preload" href="apple-icon.png" as="image">
```

Preload App Manifest and icons for Safari

index.html

```
<link rel="manifest" href="app.webmanifest">
<link rel="preload" href="app.webmanifest" as="manifest">

<link rel="apple-touch-icon" href="apple-icon.png">
<link rel="preload" href="apple-icon.png" as="image">
```

Safari ignores as="manifest" and while it downloads the icon, it ignores it for the PWA icon when installing it

So: until it supports it, **we'll ignore this solution and create a hack**.

Preload App Manifest for Safari

Using fetch API and data URI

script.js

Preload App Manifest for Safari

Using fetch API and data URI

script.js

```
fetch("manifest.json")
    .then(r=>r.text())
    .then(manifest =>


    )
```

Preload App Manifest for Safari

Using fetch API and data URI

script.js

```
fetch("manifest.json")
    .then(r=>r.text())
    .then(manifest =>
       document.querySelector("link[rel=manifest]")

    )
```

Preload App Manifest for Safari

Using fetch API and data URI

script.js

```
fetch("manifest.json")
    .then(r=>r.text())
    .then(manifest =>
        document.querySelector("link[rel=manifest]").href =
            `data:application/manifest+json;charset=utf-8,${manifest}`
    )
```

Preload App Manifest for Safari

Using fetch API and data URI

script.js

```
if ('standalone' in navigator) {     // Safari on iOS/iPadOS
    fetch("manifest.json")
        .then(r=>r.text())
        .then(manifest =>
            document.querySelector("link[rel=manifest]").href =
                `data:application/manifest+json;charset=utf-8,${manifest}`
        )
}
```

Preload Icon for Safari

Using fetch API and data URI with base64

script.js

Preload Icon for Safari

Using fetch API and data URI with base64

script.js

```
if ('standalone' in navigator) {     // Safari on iOS/iPadOS
    fetch("icon.png")
        .then(r=>r.blob())









}
```

Preload Icon for Safari

Using fetch API and data URI with base64

script.js

```
if ('standalone' in navigator) {     // Safari on iOS/iPadOS
    fetch("icon.png")
        .then(r=>r.blob())
        .then(blob => {
            let reader = new FileReader();



            reader.readAsDataURL(blob);
        })
}
```

Preload Icon for Safari

Using fetch API and data URI with base64

script.js

```
if ('standalone' in navigator) {     // Safari on iOS/iPadOS
    fetch("icon.png")
        .then(r=>r.blob())
        .then(blob => {
            let reader = new FileReader();
            reader.onload = () =>
                document.querySelector("link[rel=apple-touch-icon]").href =
                    reader.result;
            reader.readAsDataURL(blob);
        })
}
```

Be careful with deprecated Safari meta tags

Deprecated home screen web app meta tag

index.html

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

Deprecated Mobile Web App Meta Tag

If the Web App Manifest is not loaded, and the meta tag is present:

- It will install an standalone experience
- But pointing to current URL, not start_url
- Also no scope is defined, so navigation will have problems
- On some versions, a different web rendering engine is used

Do not insert the deprecated meta tag in your HTML

Summary

To improve iOS and iPadOS installability experience

- Preload and inline the Web App Manifest
- Preload and inline Apple's icon
- Do not use deprecated meta tags

Summary

Installing the Application

- Installation Architecture Review
- Track Installation for Analytics
- Promote Installation from UI
- Prioritize Store App over Browser Installation
- Improve Reliability on iOS and iPadOS

Up Next:
Managing App's Lifecycle