Progressive Web Apps - PWA

Presented by
Tran Duy Minh

TABLE OF CONTENTS

01

Definition

TWO TYPES OF APP

Web Apps

Operate universally with a **single codebase**, running across various **browsers** and devices that support those browsers.

Platform-specific Apps

Function only on their respective **device platforms,** requiring a **separate codebase** for each platform and possibly a specialized development team.

Progressive Web Apps

A traditional web app that is **progressively enhanced** using open web technologies, to make sure it delivers the **best possible experience** on every device, based on available capabilities.

This allows Progressive Web Apps to combine the **reach** (broad availability and access) of website experiences with the **capability** (hardware features and rich resources) of platform-specific experiences.

Progressive Enhancement

Older Devices/Browsers

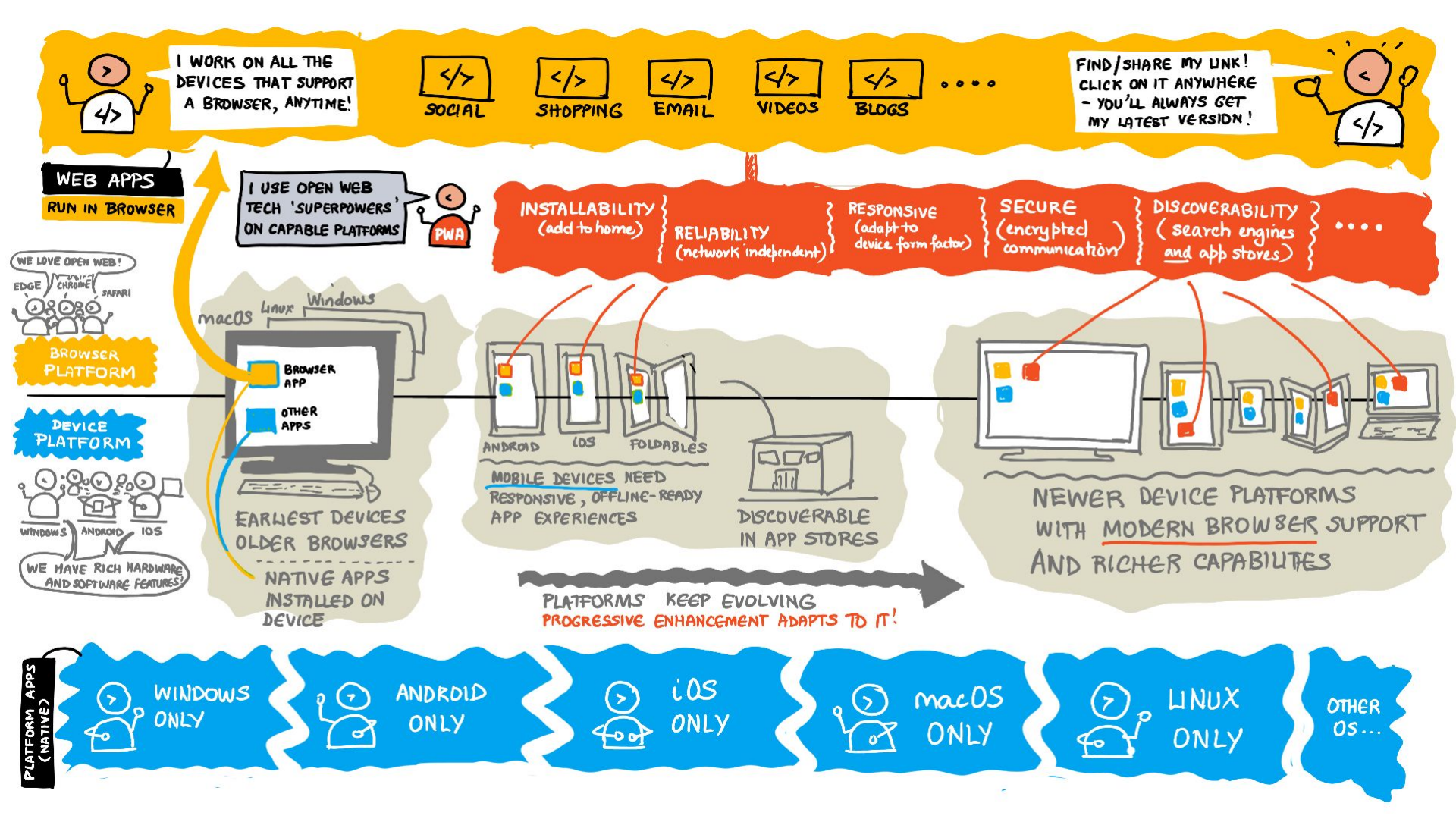
Where nothing new may be detected, PWA delivers a **baseline website experience**.

Newer devices

PWA can detect device form factors and deliver **responsive experiences** that align with platform-specific behaviors.

Modern browsers

PWA support features like **installability** and **offline operation** - just like platform-specific apps


I WORK ON ALL THE DEVICES THAT SUPPORT A BROWSER, ANYTIME!
SOCIAL
SHOPPING
EMAIL
VIDEOS
BLOGS
....
FIND/SHARE MY LINK! CLICK ON IT ANYWHERE - YOU'LL ALWAYS GET MY LATEST VERSION!
WEB APPS RUN IN BROWSER
I USE OPEN WEB TECH 'SUPERPOWERS' ON CAPABLE PLATFORMS
PWA
INSTALLABILITY (add to home)
RELIABILITY (network independent)
RESPONSIVE (adapt to device form factor)
SECURE (encrypted communication)
DISCOVERABILITY (search engines and app stores)
....
WE LOVE OPEN WEB!
EDGE
CHROME
SAFARI
BROWSER PLATFORM
DEVICE PLATFORM
WINDOWS
ANDROID
IOS
WE HAVE RICH HARDWARE AND SOFTWARE FEATURES!
macOS
Linux
Windows
BROWSER APP
OTHER APPS
EARLIEST DEVICES OLDER BROWSERS
NATIVE APPS INSTALLED ON DEVICE
ANDROID
iOS
FOLDABLES
MOBILE DEVICES NEED RESPONSIVE, OFFLINE-READY APP EXPERIENCES
DISCOVERABLE IN APP STORES
NEWER DEVICE PLATFORMS WITH MODERN BROWSER SUPPORT AND RICHER CAPABILITIES
PLATFORMS KEEP EVOLVING
PROGRESSIVE ENHANCEMENT ADAPTS TO IT!
PLATFORM APPS (NATIVE)
WINDOWS ONLY
ANDROID ONLY
iOS ONLY
macOS ONLY
LINUX ONLY
OTHER OS...

Origin

In 2007, alongside the launch of the iPhone, Steve Jobs introduced the visionary idea of **Web Apps,** emphasizing app development around Safari's capabilities and downplaying the need for native apps.

However, his enthusiasm waned, and four months later, an SDK for building more efficient native iOS apps was announced.

Origin

In 2015, Chrome developer **Alex Russell** and designer **Frances Berriman** introduced the term "Progressive Web Apps" in an article that called for a fundamental shift in our tools and understanding to "build better experiences across devices and contexts with a single codebase."

02

Characteristics

PWA’s Operation

Access

Customers **access any website** through the browser on their mobile devices. If the website supports PWA features, users will be prompted to install the site's PWA.

Install

If users agree, the browser will **download and store** the PWA on their device.

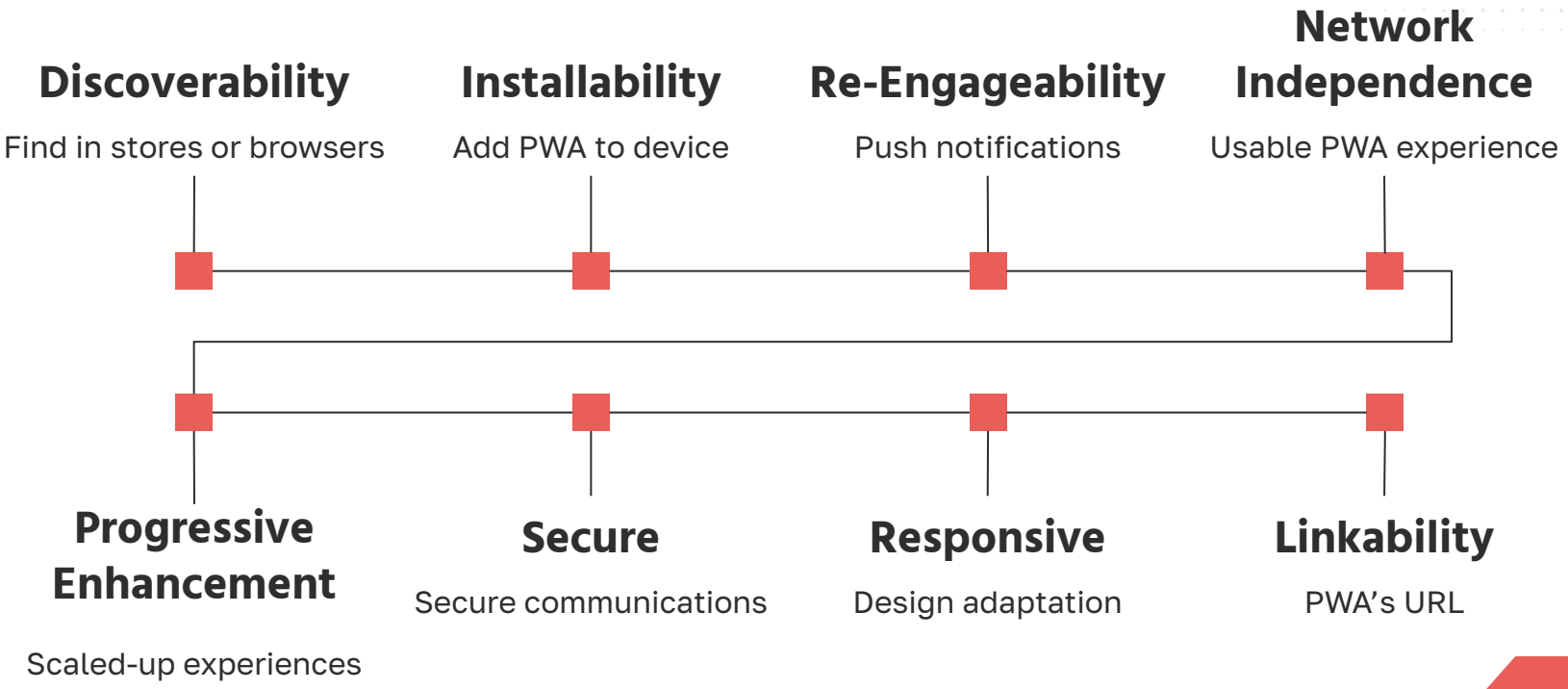
Open

When users want to access the website, they can open **the site's PWA** instead of the browser

Offline

If the PWA is designed to work offline, users can still access the website. When the network connection is restored, the PWA will update with the latest data from the website and synchronize the data on the user's device.

PWA’s Characteristics



03

Comparison & Usage

PWA vs Web App

Installation

Offline capability

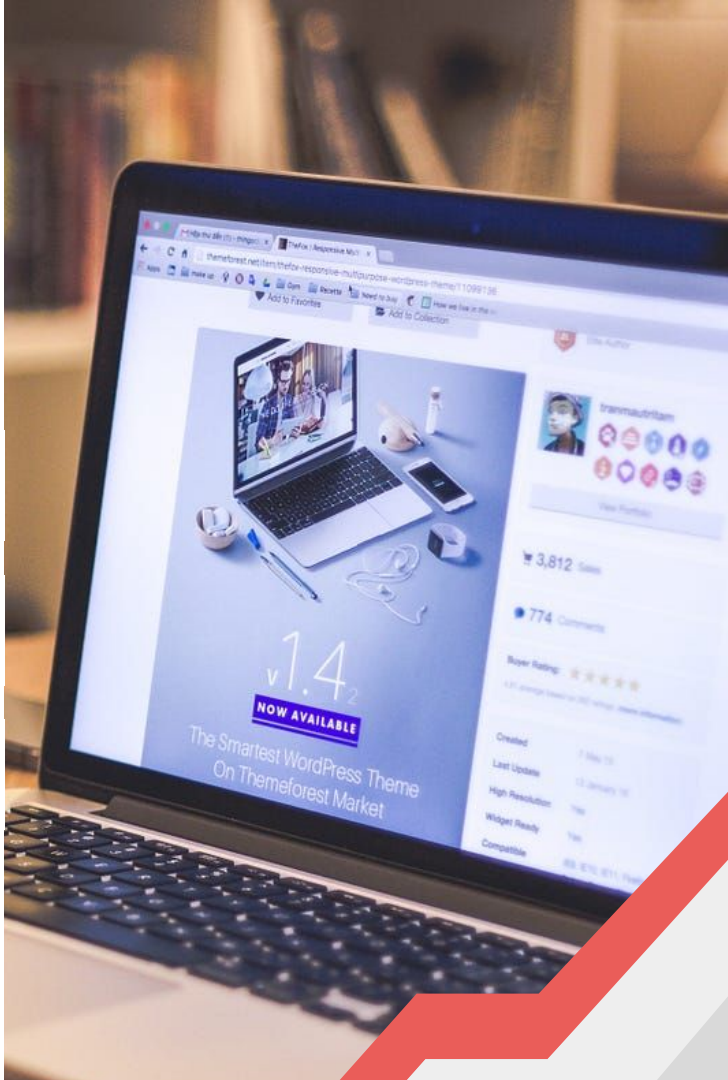
Push notifications

Performance

Security

Customization

Distribution



PWA vs Native App

Development process

Building a PWA is **cheaper and faster** than a native app, but native apps benefit from app store **reliability and promotion**.

Security

PWAs ensure secure data exchange with **HTTPS**, while native apps offer **built-in security measures** and higher user trust due to App Store requirements.

Installation

A PWA requires **no App Store** or installation, taking up less space and always showing the **latest version**, while offering home screen access, notifications, and sharing via URL.

Performance

PWAs **load faster** with service workers, but native apps have **lower latency and better performance** by accessing device hardware.

Functionality

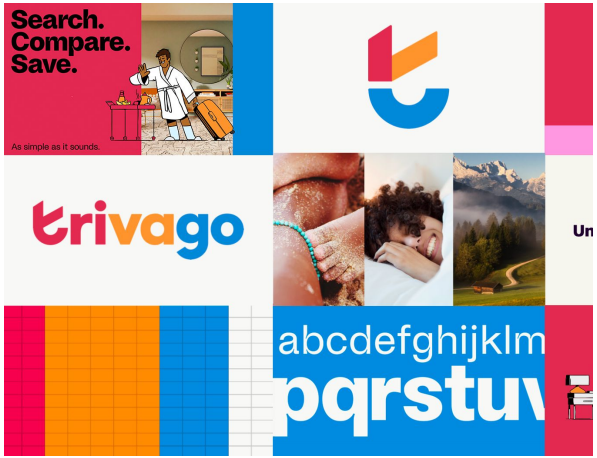
PWAs access device features but are **limited by browser restrictions**, while native apps have full access to all device features and advanced functionalities.

When to PWA?

One reason brands develop native apps is to cater to users who return to their websites to **perform specific actions frequently**. Apps make it easier for these functions to be performed without going to the brand's website.

Other times you should consider using a progressive web app are when:

- You don't have the budget for a full-fledged app.
- You need to get to market fast.
- Proper indexing on search engines is crucial.
- Cross-platform compatibility is essential to your business.
- You need to reach a wider audience.



Case Studies

How Leading Brands Harness PWAs to Drive Their Success

04

PWA’s Building Blocks

Three Core Building Blocks

HTTPS 

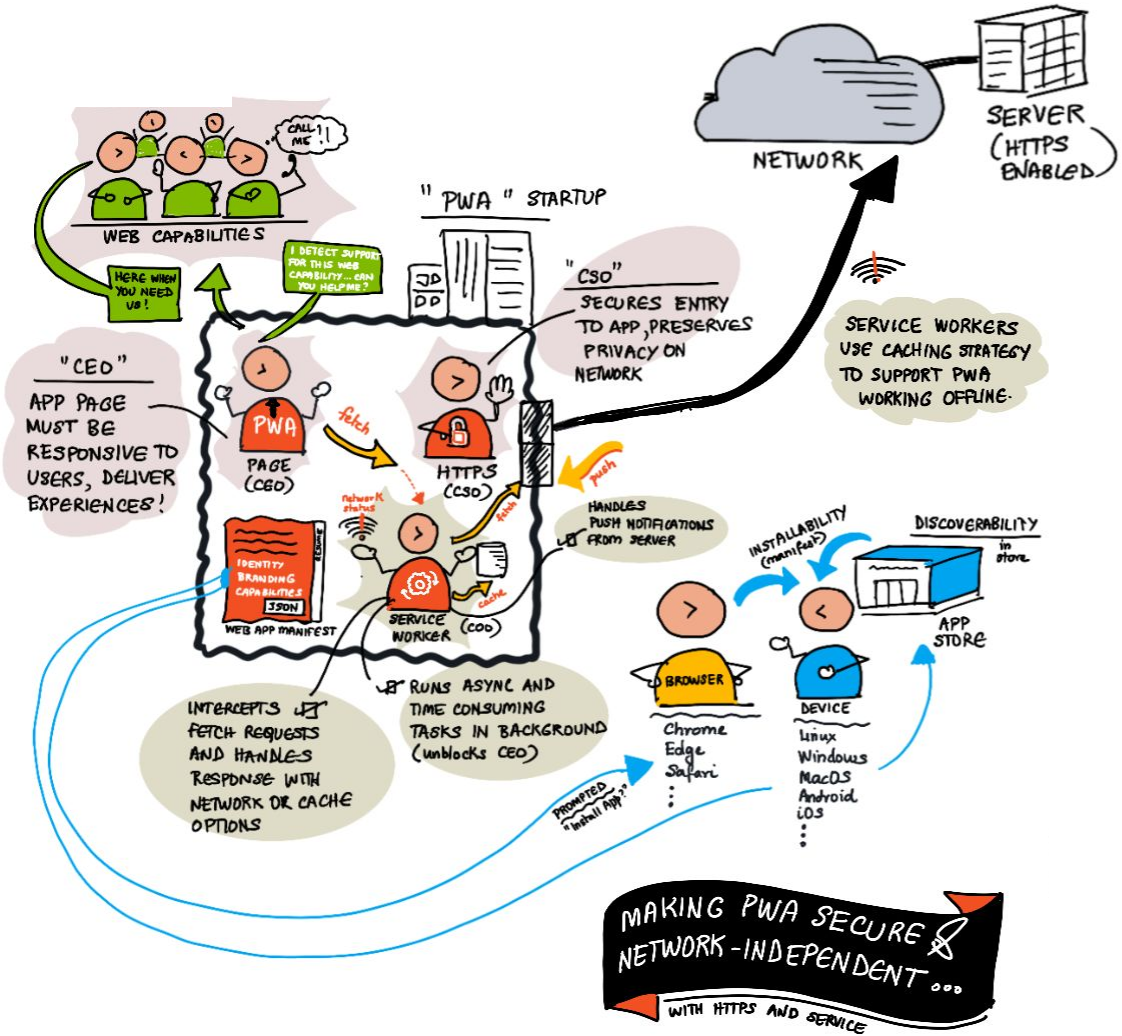
Makes your PWA secure

Web App Manifest 

Makes your PWA installable

Service Workers 

Makes your PWA reliable and network-independent



PWA's Building blocks

HTTPS

HTTPS or ***HyperText Transfer Protocol Secure*** is a secure version of the HTTP protocol, encrypting end-to-end communications between client and server endpoints in your web app by default.

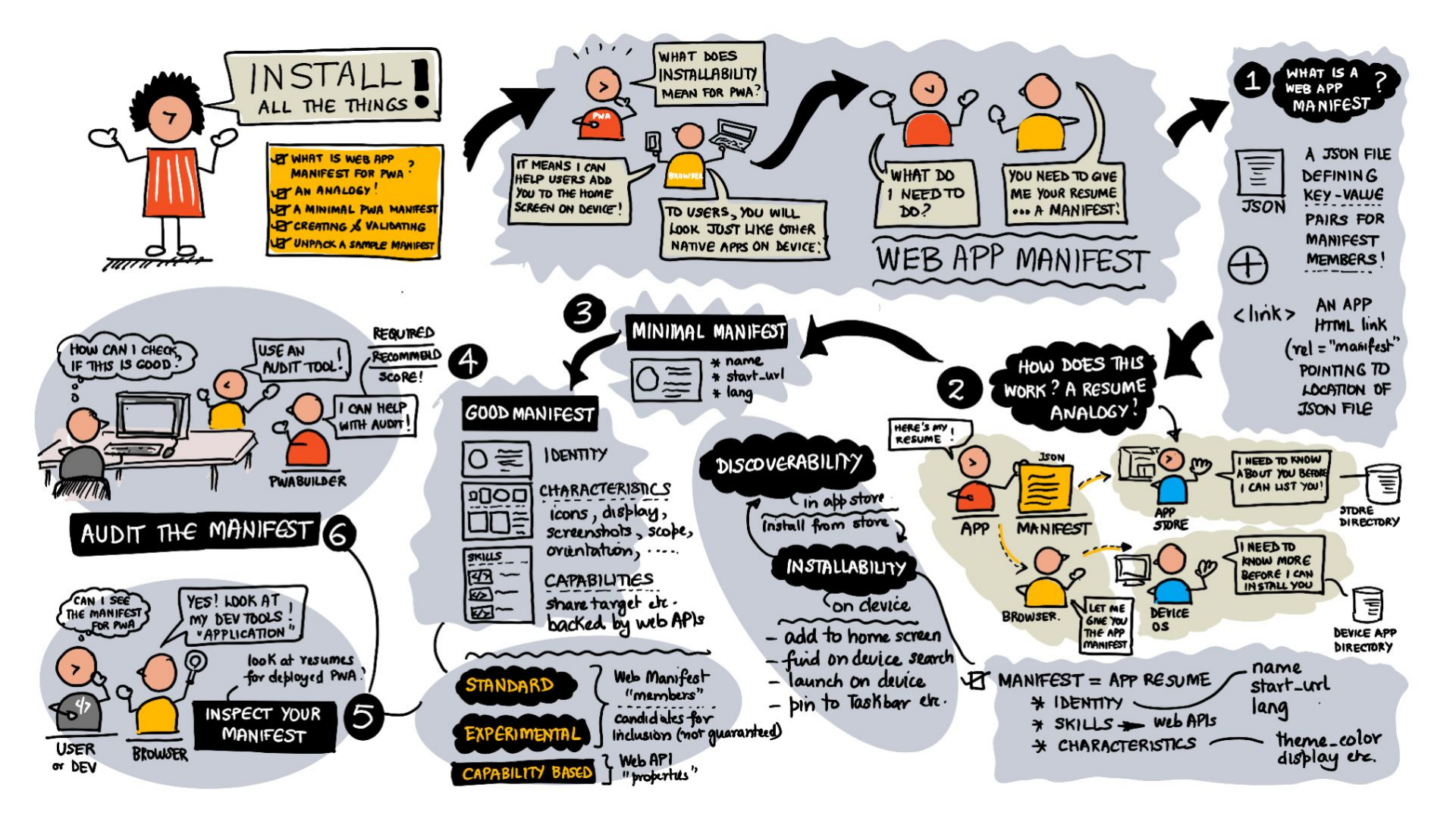
Progressive Web Apps **must be served** from an HTTPS endpoint to ensure secure communications, provide user privacy safeguards and guarantee content authenticity. HTTPS is mandatory for Service Workers - the core PWA technology required for reliable, offline-friendly operation.

Web App Manifest

The **Web App Manifest** is an open web specification of a JSON format that is critical to making PWAs installable.

Functionally, it *governs how your PWA **looks and behaves** when installed on device* by defining the properties (key-value pairs) that characterize its appearance and behaviors.

```
{
    "name": "DevTools Tips",
    "short_name": "DevTools Tips",
    "start_url": "/",
    "categories": [
        "productivity",
        "devtools",
    ],
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#ffffff",
    "scope": "/",
    "description": "A collection of useful
cross-browser DevTools tips",
    "icons": [
        {
            "src": "/assets/logo-192.png",
            "sizes": "192x192",
            "type": "image/png"
        }
    ],
    "screenshots": [
        {
            "src": "/assets/screenshot-home.png",
            "sizes": "1992x1773",
            "type": "image/png"
        },
    ],
    "url_handlers": [
        {
            "origin": "https://devtoolstips.org"
        }
    ]
}
```


INSTALL ALL THE THINGS!
WHAT IS WEB APP MANIFEST FOR PWA?
AN ANALOGY!
A MINIMAL PWA MANIFEST
CREATING & VALIDATING
UNPACK A SAMPLE MANIFEST
WHAT DOES INSTALLABILITY MEAN FOR PWA?
PWA
IT MEANS I CAN HELP USERS ADD YOU TO THE HOME SCREEN ON DEVICE!
BROWSER
TO USERS, YOU WILL LOOK JUST LIKE OTHER NATIVE APPS ON DEVICE!
WHAT DO I NEED TO DO?
YOU NEED TO GIVE ME YOUR RESUME ... A MANIFEST!
WEB APP MANIFEST
1 WHAT IS A WEB APP MANIFEST?
JSON
A JSON FILE DEFINING KEY-VALUE PAIRS FOR MANIFEST MEMBERS!
<link>
AN APP HTML link (rel="manifest" POINTING TO LOCATION OF JSON FILE
2 HOW DOES THIS WORK? A RESUME ANALOGY!
HERE'S MY RESUME!
APP
JSON
MANIFEST
APP STORE
I NEED TO KNOW ABOUT YOU BEFORE I CAN LIST YOU!
STORE DIRECTORY
BROWSER
LET ME GIVE YOU THE APP MANIFEST
DEVICE OS
I NEED TO KNOW MORE BEFORE I CAN INSTALL YOU
DEVICE APP DIRECTORY
3 MINIMAL MANIFEST
* name
* start_url
* lang
DISCOVERABILITY
in app store
install from store
INSTALLABILITY
on device
- add to home screen
- find on device search
- launch on device
- pin to Taskbar etc.
MANIFEST = APP RESUME
* IDENTITY
name
start_url
lang
* SKILLS → web APIs
* CHARACTERISTICS
theme_color
display etc.
4 GOOD MANIFEST
IDENTITY
CHARACTERISTICS
icons, display, screenshots, scope, orientation, ...
SKILLS
CAPABILITIES
share target etc. backed by web APIs
STANDARD
Web Manifest "members"
EXPERIMENTAL
candidates for inclusion (not guaranteed)
CAPABILITY BASED
Web API "properties"
5 INSPECT YOUR MANIFEST
look at resumes for deployed PWA!
CAN I SEE THE MANIFEST FOR PWA
YES! LOOK AT MY DEV TOOLS! "APPLICATION"
USER or DEV
BROWSER
6 AUDIT THE MANIFEST
HOW CAN I CHECK IF THIS IS GOOD?
USE AN AUDIT TOOL!
I CAN HELP WITH AUDIT!
PWABUILDER
REQUIRED
RECOMMEND
SCORE!

Web App Manifest

Creating a manifest

- Create a *manifest.json* and populate its properties.
- Linking it to app HTML to advertise your PWA status.

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

A suggested minimal manifest should have at least these three - where ***start_url*** defines the entry point (default path shown) when app is launched on device.

```
{
    "name": "My Sample PWA",
    "lang": "en-US",
    "start_url": "/"
}
```

Supported members

Description of some supported members in the manifest:

- **short_name** - app name for constrained spaces (e.g., home screen)
- **categories** - hints for stores or app catalogs.
- **display** - how much of the browser UI does the user see in the app?
- **background_color** - placeholder to show (before stylesheet loads)
- **theme_color** - default theme color for app
- **scope** - what web pages can be viewed (navigation scope)
- **description** - what is the app about?
- **icons** - array of icon images used in various contexts
- **screenshots** - showcase application UI/UX (e.g., for app stores)

Service Worker

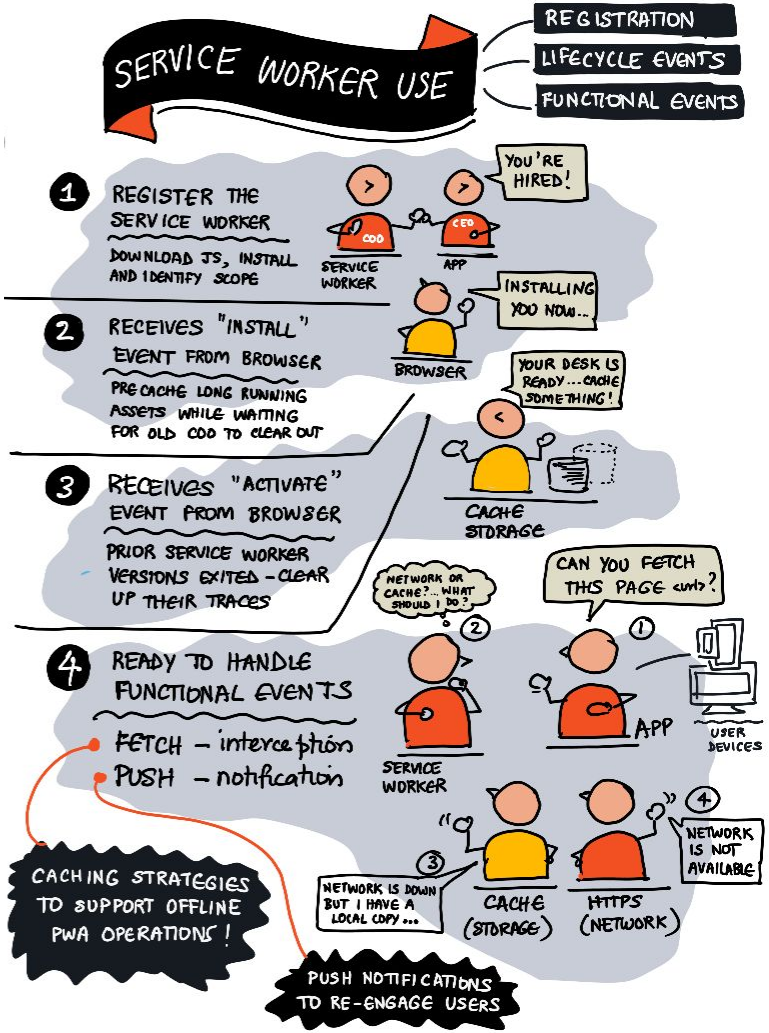
Service Workers are a special type of **Web Worker**. Web Workers operate in a separate thread, allowing them to execute long-running or asynchronous tasks in the background, minimizing the impact on page performance.

Reliable 

Deliver **usable experiences** even under flaky or offline network conditions. Service worker do this by *intercepting* network fetch requests from the page and *strategically* handling them.

Re-Engageable 

Having the ability to **alert users** to app changes or context. Service worker do this by listening for asynchronous push notifications (from a server) and working with platform capabilities to deliver alerts.



Lifecycle & Functional events

Lifecycle Events

Service worker registration includes three phases. Once that is complete, the service worker is ready to listen for lifecycle events (install, activate) to set itself up for success.

1. Registration: The browser registers the service worker, kicking off the Service Worker lifecycle.

2. Installation: The browser triggers **install** as the first event to the Service Worker. It can use this for pre-caching resources (e.g., populate cache with long-lived resources like logos or offline pages).

```
self.addEventListener( "install", function( event ){
    console.log( "WORKER: install event in progress." );
});
```

Lifecycle Events

Service worker registration includes three phases. Once that is complete, the service worker is ready to listen for lifecycle events (install, activate) to set itself up for success.

3. Activation:

The browser sends the **activate** event to indicate that the service worker has been installed. This service worker can now clean up actions (e.g., remove old caches from prior versions) and ready itself to handle *functional* events.

If there is an old service worker in play, you can use **clients.claim()** to immediately replace the old service worker with your new one.

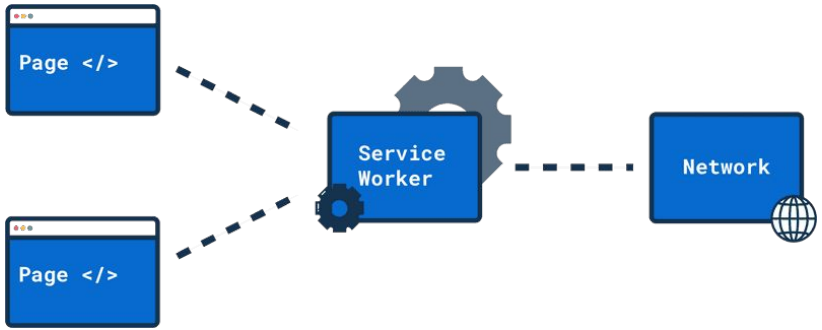
```
self.addEventListener( "activate", function( event ){
    console.log( "WORKER: activation event in progress." );
    clients.claim();
    console.log( "WORKER: all clients are now controlled" );
});
```

Functional Events

Functional events are those that require the asynchronous or background processing abilities of service workers to support reliable and re-enageable behaviors. For now, think about just two: “**fetch**” and “**push**”.

The **fetch event** is triggered when the browser tries to access a page that lies within the *scope* of the service worker. The service worker acts as an interceptor - returning a response either from the cache or from the network (or some combination of both) based on predefined strategies

```
self.addEventListener( "fetch", function( event ){
    console.log( "WORKER: Fetching", event.request );
});
```

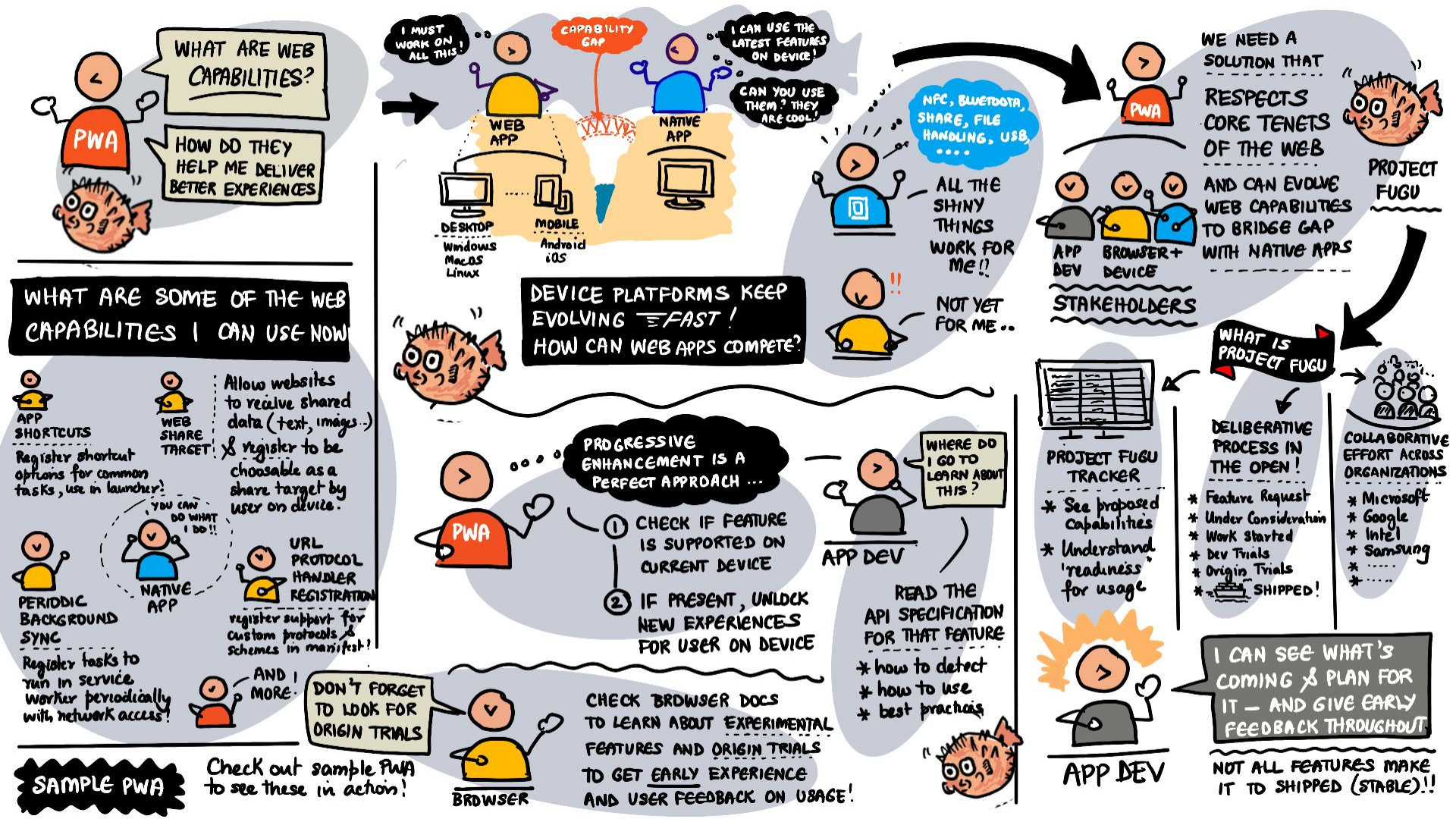


Functional Events 

Functional events are those that require the asynchronous or background processing abilities of service workers to support reliable and re-enageable behaviors. For now, think about just two: “**fetch**” and “**push**”.

The **push event** is triggered when the browser receives a push message from a server to display as a toast notification to users. This occurs only if the PWA had previously *subscribed* for server notifications *and the user* has granted the PWA permission to receive them. Push events are critical to **re-engaging** users when the app is not otherwise active.

```
self.addEventListener( "push", function( event ){
    console.log( "WORKER: Received notification",
event.data );
});
```


WHAT ARE WEB CAPABILITIES?
PWA
HOW DO THEY HELP ME DELIVER BETTER EXPERIENCES
I MUST WORK ON ALL THIS!
CAPABILITY GAP
I CAN USE THE LATEST FEATURES ON DEVICE!
CAN YOU USE THEM? THEY ARE COOL.
WEB APP
NATIVE APP
DESKTOP
Windows
MacOS
Linux
MOBILE
Android
iOS
NFC, BLUETOOTH, SHARE, FILE HANDLING, USB ....
ALL THE SHINY THINGS WORK FOR ME!!
NOT YET FOR ME..
DEVICE PLATFORMS KEEP EVOLVING FAST! HOW CAN WEB APPS COMPETE?
PWA
WE NEED A SOLUTION THAT
RESPECTS CORE TENETS OF THE WEB
AND CAN EVOLVE WEB CAPABILITIES TO BRIDGE GAP WITH NATIVE APPS
PROJECT FUGU
APP DEV
BROWSER + DEVICE
STAKEHOLDERS
WHAT ARE SOME OF THE WEB CAPABILITIES I CAN USE NOW
APP SHORTCUTS
Register shortcut options for common tasks, use in launcher!
WEB SHARE TARGET
Allow websites to receive shared data (text, images..) & register to be choosable as a share target by user on device.
PERIODIC BACKGROUND SYNC
Register tasks to run in service worker periodically with network access!
YOU CAN DO WHAT I DO!!
NATIVE APP
URL PROTOCOL HANDLER REGISTRATION
register support for custom protocols & schemes in manifest!
AND MORE
SAMPLE PWA
Check out sample PWA to see these in action!
PWA
PROGRESSIVE ENHANCEMENT IS A PERFECT APPROACH ...
1 CHECK IF FEATURE IS SUPPORTED ON CURRENT DEVICE
2 IF PRESENT, UNLOCK NEW EXPERIENCES FOR USER ON DEVICE
APP DEV
WHERE DO I GO TO LEARN ABOUT THIS?
READ THE API SPECIFICATION FOR THAT FEATURE
* how to detect
* how to use
* best practices
DON'T FORGET TO LOOK FOR ORIGIN TRIALS
BROWSER
CHECK BROWSER DOCS TO LEARN ABOUT EXPERIMENTAL FEATURES AND ORIGIN TRIALS TO GET EARLY EXPERIENCE AND USER FEEDBACK ON USAGE!
WHAT IS PROJECT FUGU
PROJECT FUGU TRACKER
* See proposed capabilities
* Understand 'readiness' for usage
DELIBERATIVE PROCESS IN THE OPEN!
* Feature Request
* Under Consideration
* Work Started
* Dev Trials
* Origin Trials
* SHIPPED!
COLLABORATIVE EFFORT ACROSS ORGANIZATIONS
* Microsoft
* Google
* Intel
* Samsung
* .....
* .....
APP DEV
I CAN SEE WHAT'S COMING & PLAN FOR IT — AND GIVE EARLY FEEDBACK THROUGHOUT
NOT ALL FEATURES MAKE IT TO SHIPPED (STABLE)!!

05

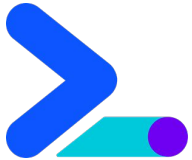
Demo

PWA in React

Bonus: Resources for learning PWA

#30DaysOfPWA

A multi-team effort from Microsoft Edge, Windows, and Developer Relations teams with published articles that aim to introduce developers to Progressive Web App



Learn PWA - web.dev

A course that breaks down every aspect of modern progressive web app development brought to you by web.dev, a site which provide content written by members of the Chrome team, and external experts.

Summary

Progressive web app

A traditional web app that is **progressively enhanced** using open web technologies, to make sure it delivers the **best possible experience** on every device, based on available capabilities.

PWA's Building blocks

HTTPS: Makes your PWA secure
Web App Manifest: Makes your PWA installable
Service Worker: Makes your PWA reliable and network-independent

PWA's Characteristics

- Discoverability
- Installability
- Re-Engageability
- Network independence
- Progressive enhancement
- Secure
- Responsive
- Linkability

Thanks!

Do you have any questions?