Programming the PWA with Web Tools

Maximiliano Firtman
MOBILE+WEB DEVELOPER

@firt www.firt.mobi

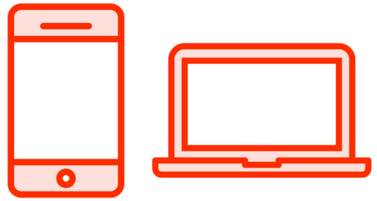
Overview

Programming the PWA with web tools

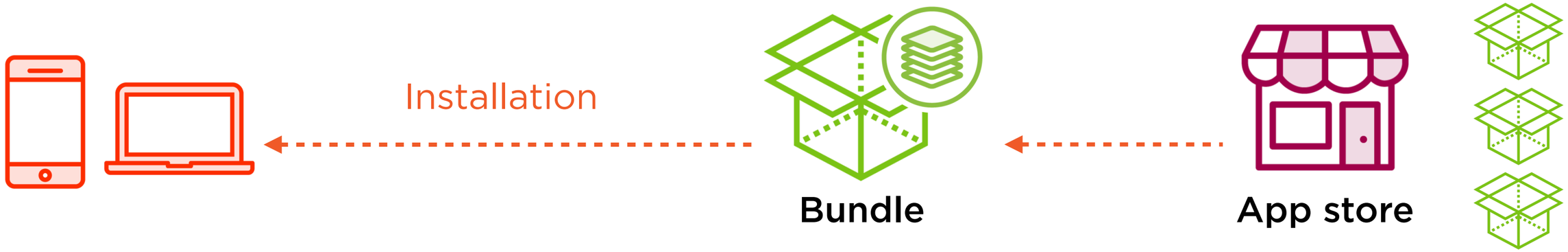
- Service Workers: the brain of a PWA
- Caching and serving resources
- Adding a Service Worker
- Doing PWAs with Angular
- Doing PWAs with React
- PWAs everywhere

Service Workers: The Brain of a PWA

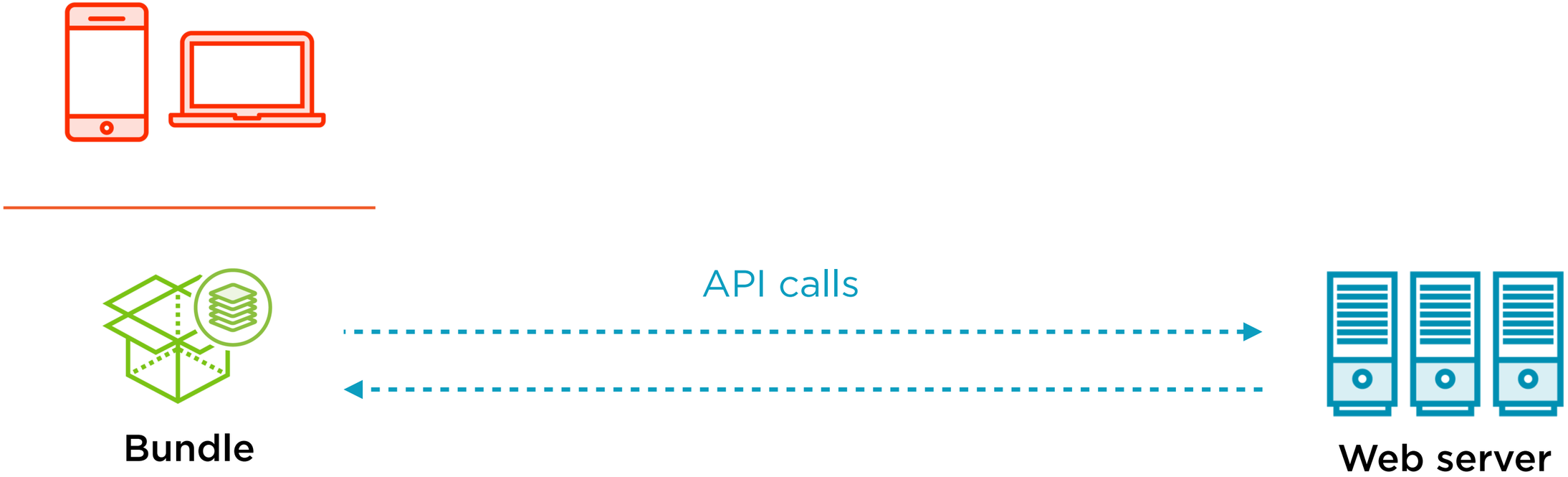
Native Apps



Native Apps



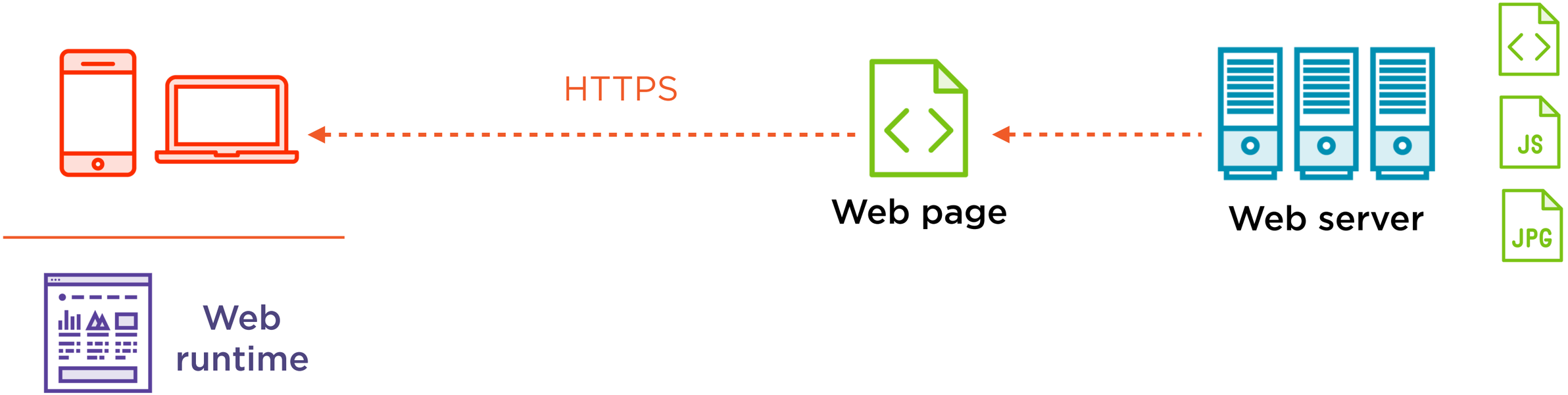
Native Apps



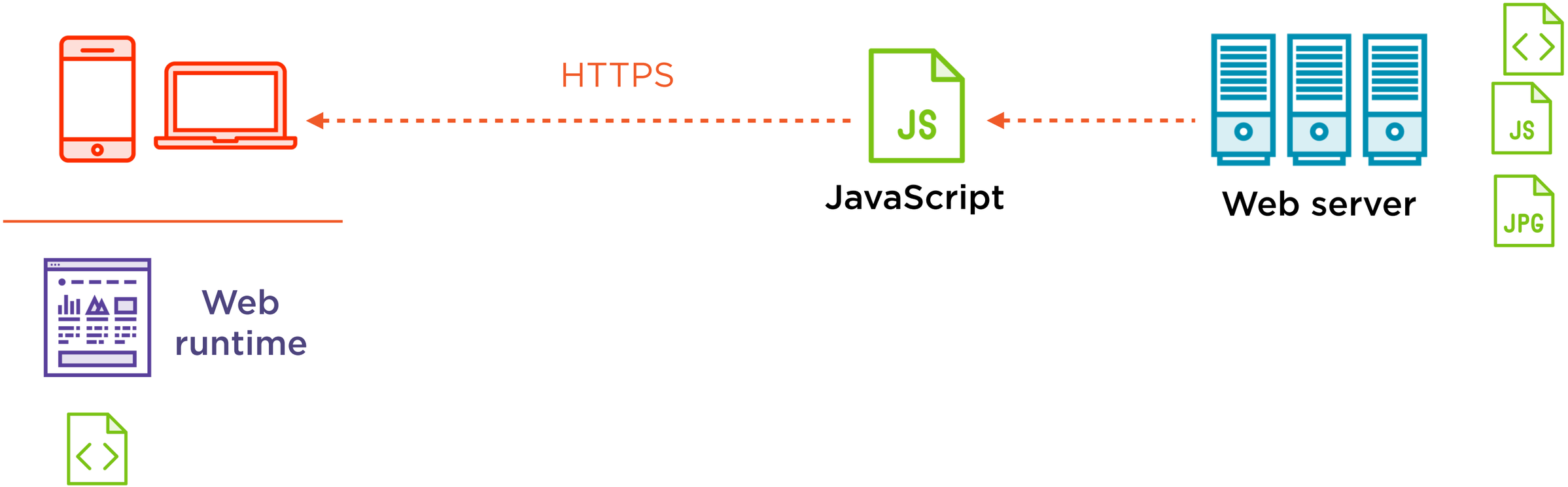
Native Apps - Offline

Websites and Web Apps

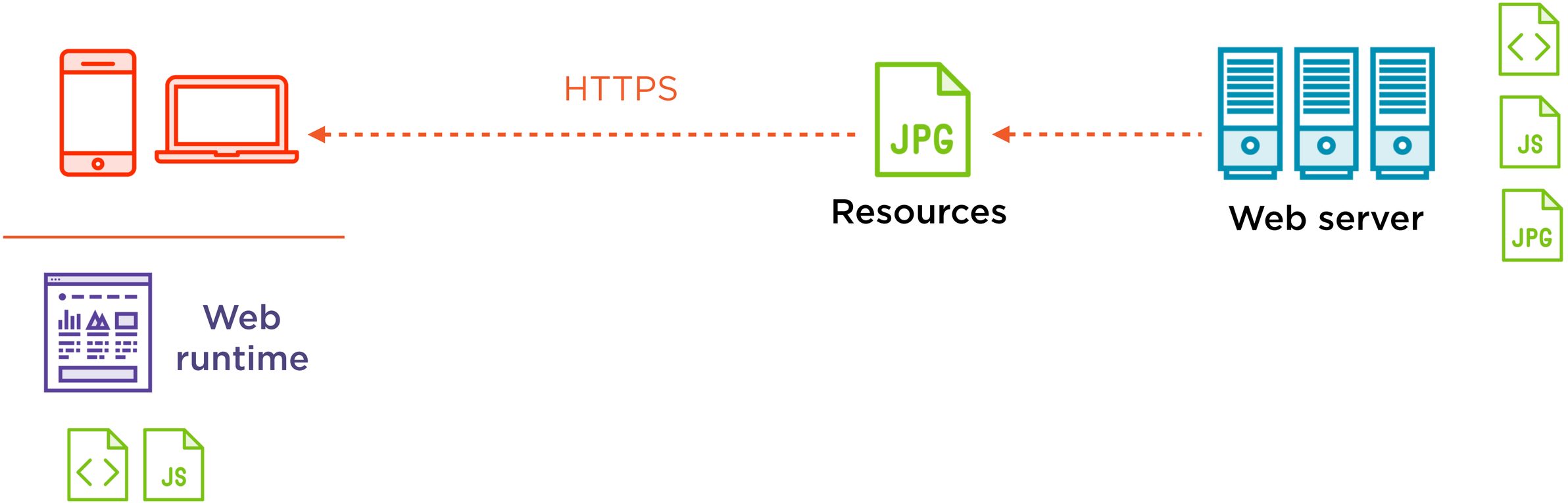
Websites and Web Apps



Websites and Web Apps

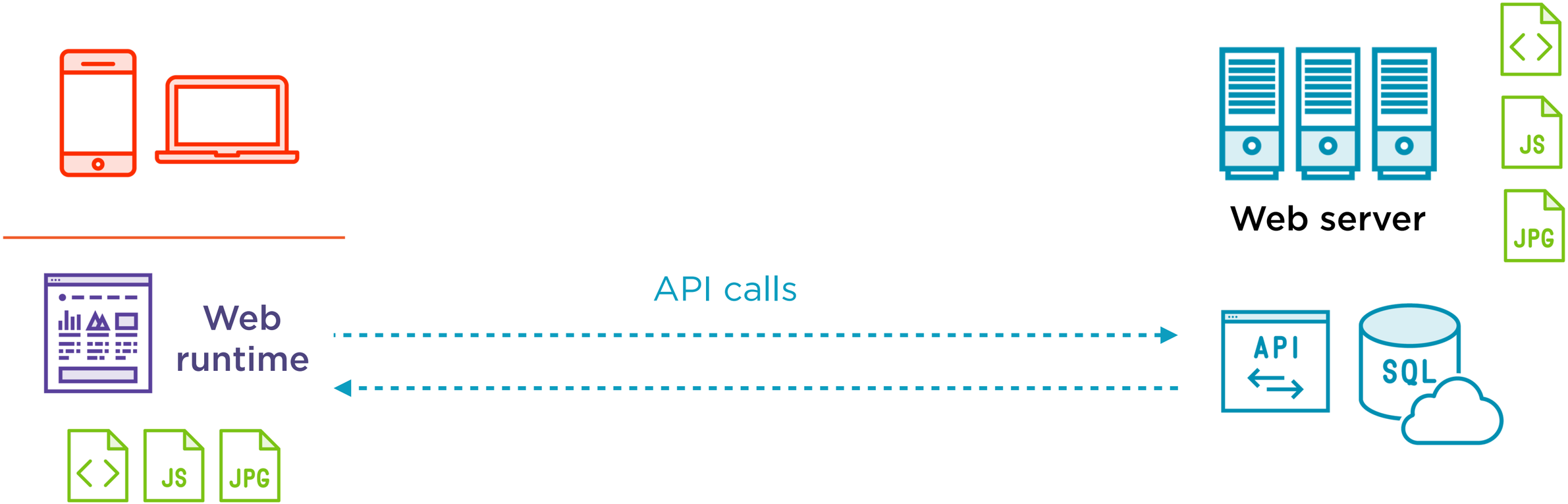


Websites and Web Apps



Websites and Web Apps

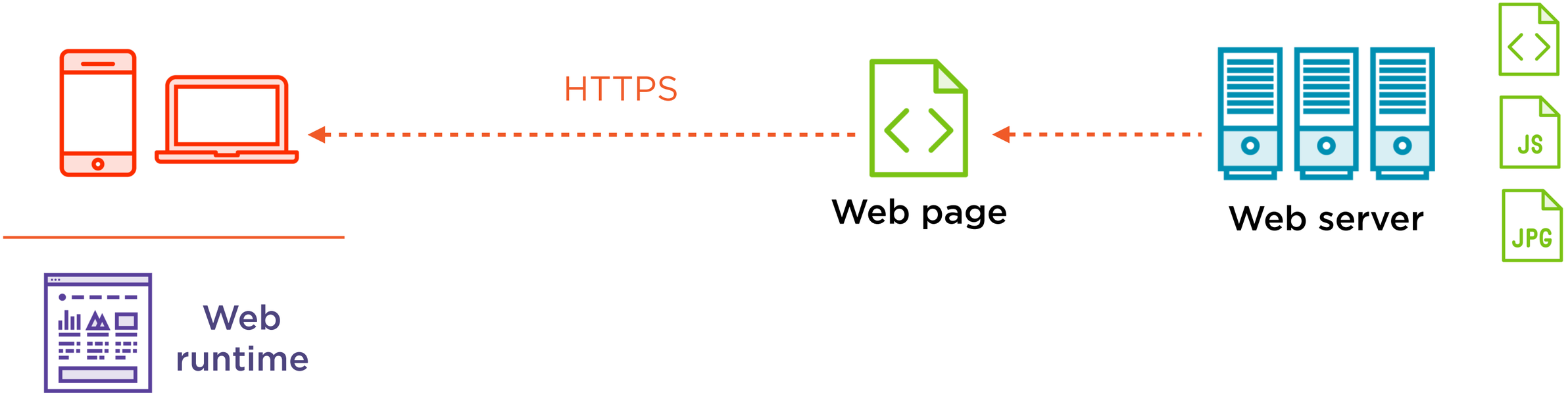
Websites and Web Apps



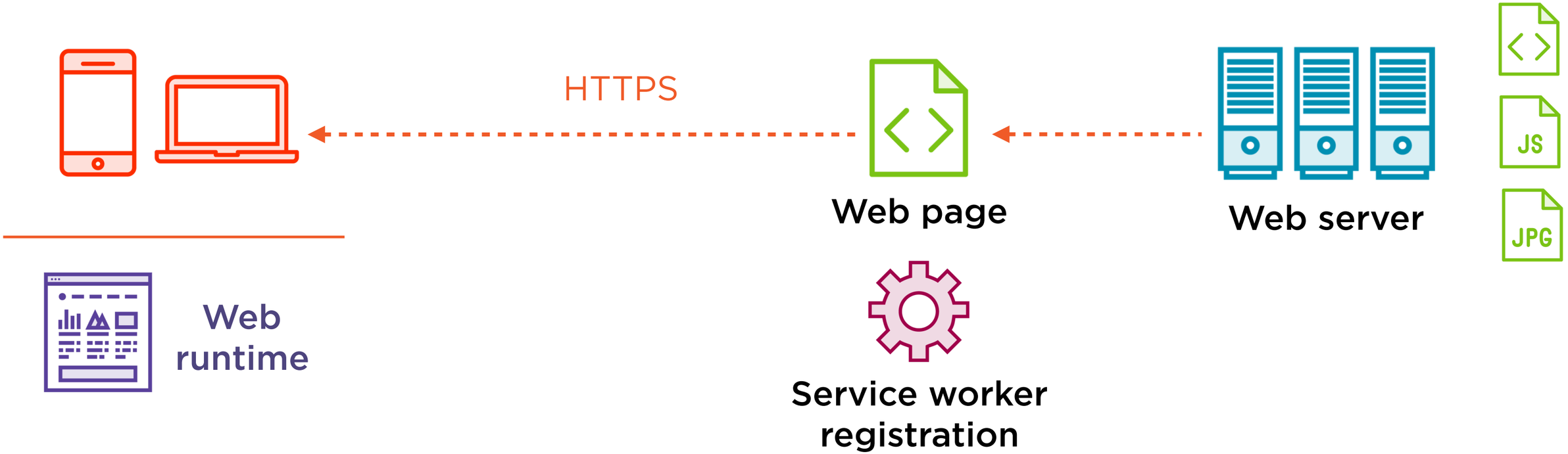
Websites and Web Apps - Offline

Progressive Web Apps

Progressive Web Apps

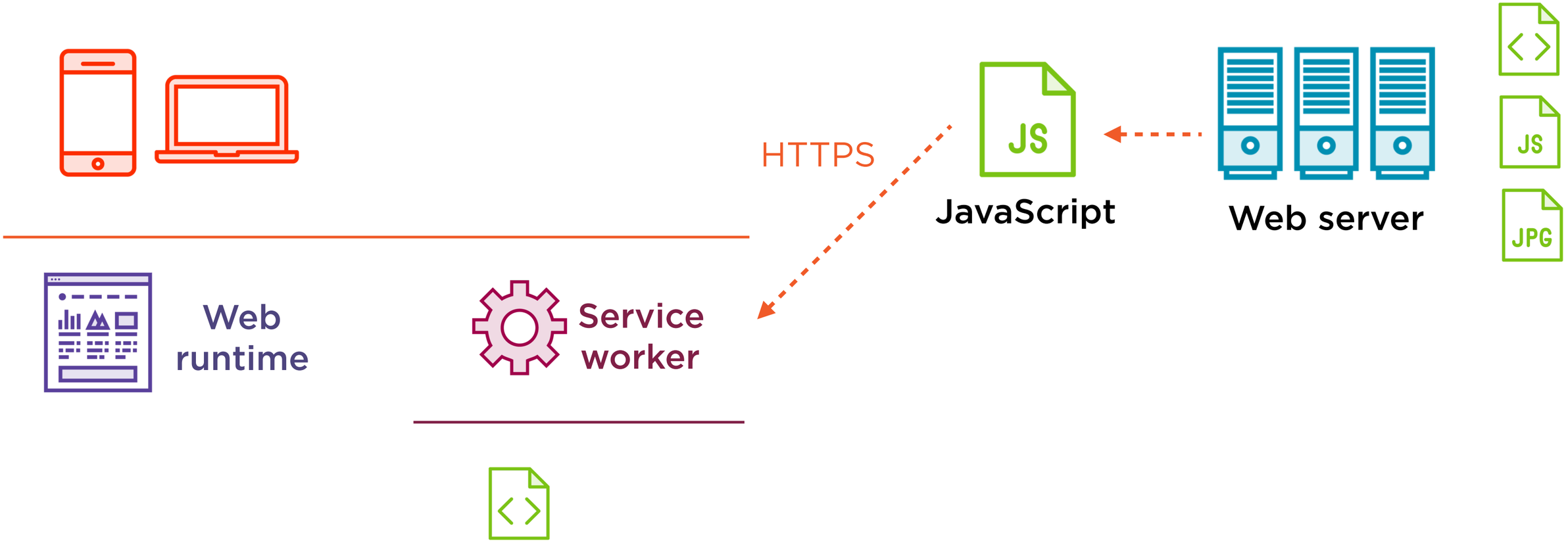


Progressive Web Apps

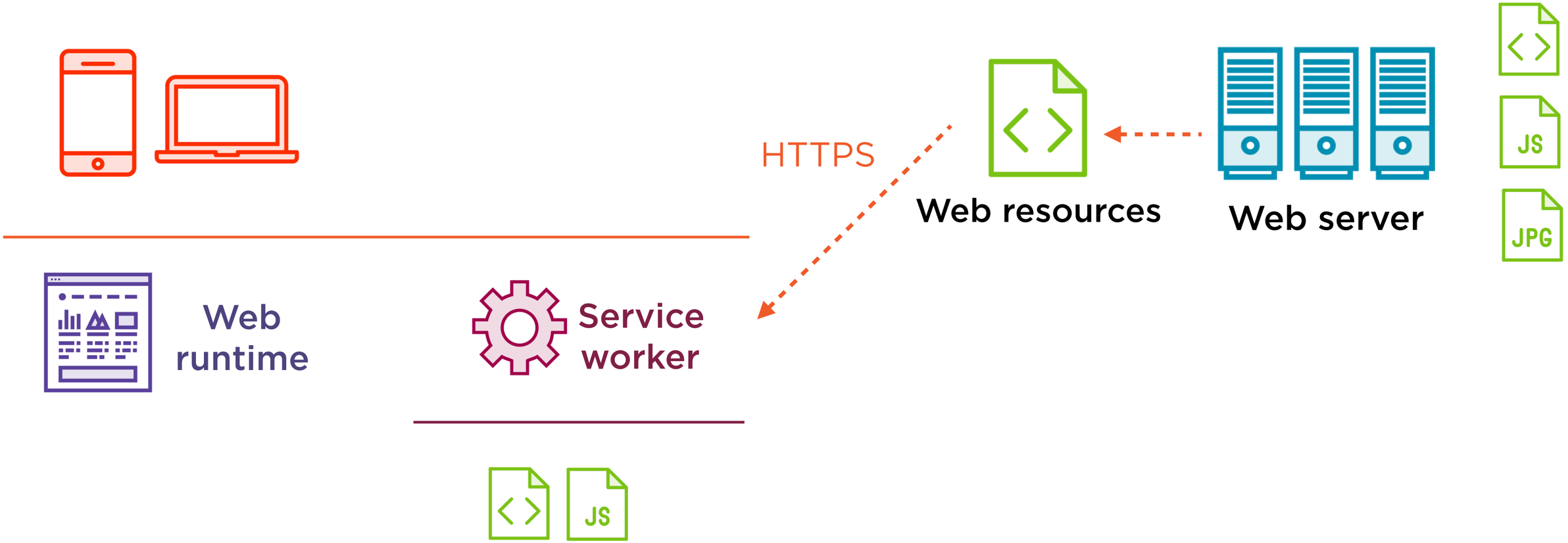


Progressive Web Apps

Progressive Web Apps



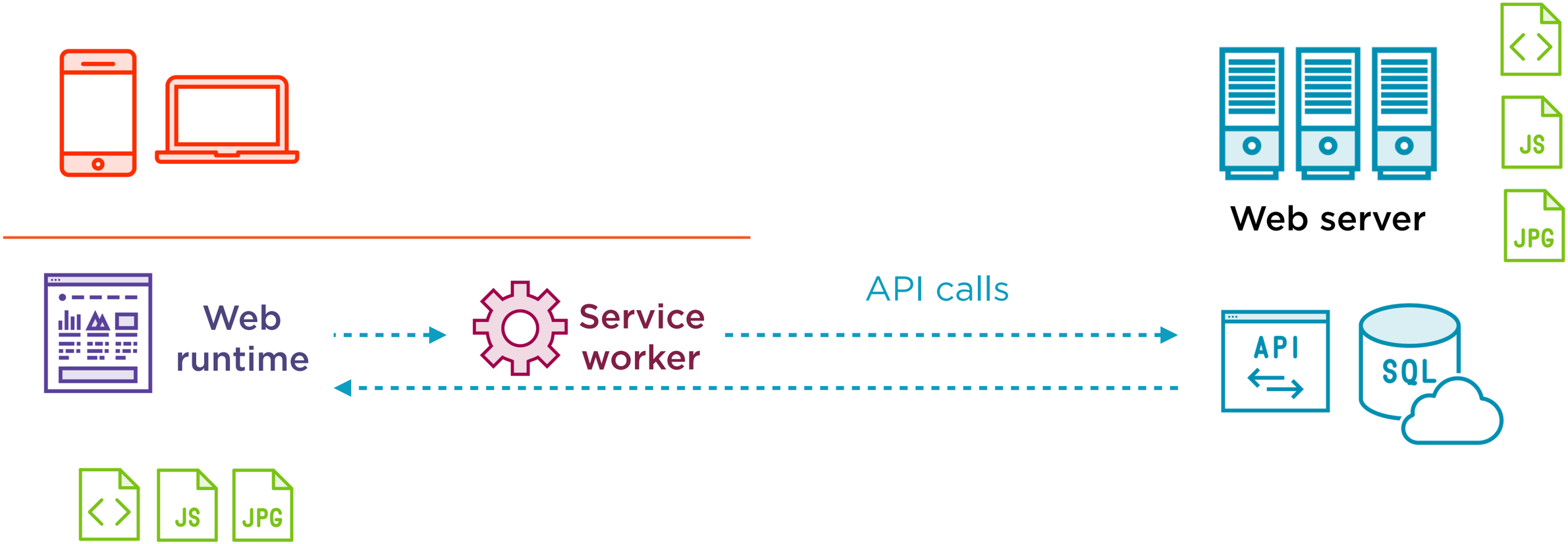
Progressive Web Apps



Progressive Web Apps

Progressive Web Apps

Progressive Web Apps

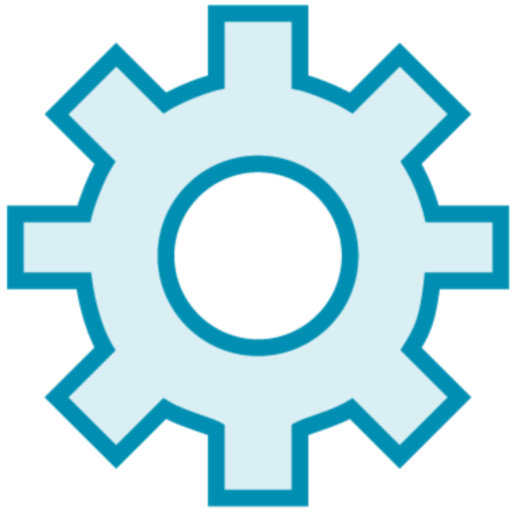


Progressive Web Apps - Offline

Service Worker

A JavaScript file running in its own thread that will act as a local installed web server or web proxy for your PWA, including resources and API calls

Service Worker



- Runs client-side in browser's engine
- HTTPS required
- Installed by a web page
- Own thread and lifecycle
- Acts as a network proxy or local web server in the name of the real server
- Abilities to run in the background
- No need for user's permission

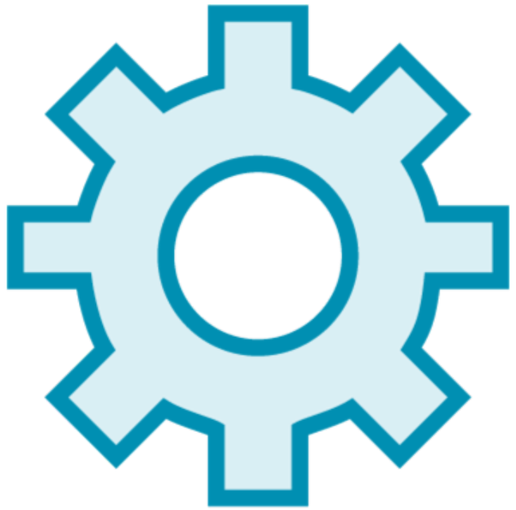
Scope

An origin (host and port) and a path

Scope

An origin (host and port) and a path
https://mydomain.com or https://domain.com/myapp

Service Worker and Scopes



It manages all pages within browser and within installed app from scope

It's installed by any page in the scope

After installed, it can serve all files requested from the scope

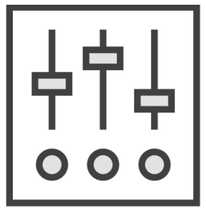
Only one service worker is allowed

WebKit adds partition management

Advantages

Fast PWA loading with local cache

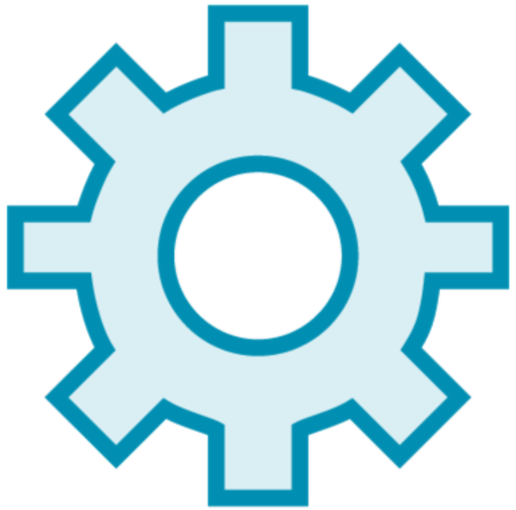
Offline support serving locally



Query the context to offer the best possible experience

Data synchronization with the server

Service Worker Extras



Use progressive enhancement for extras

Background sync

Background fetch

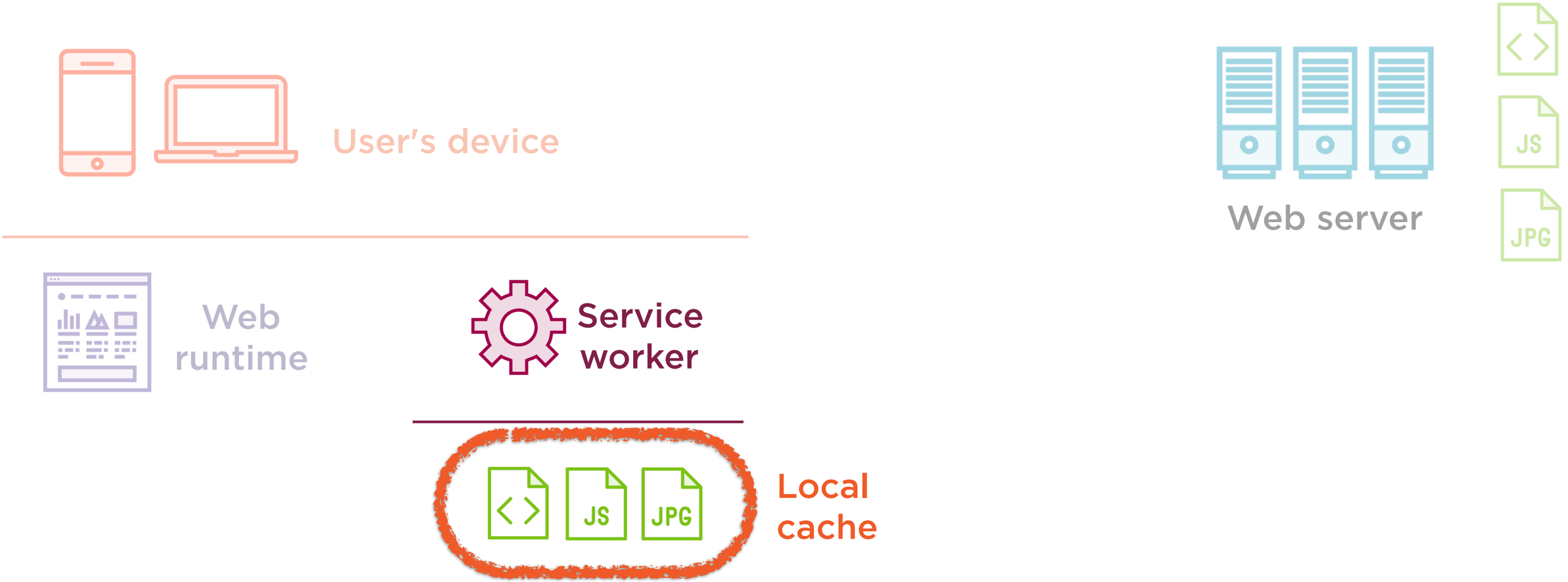
Web push

Future ideas

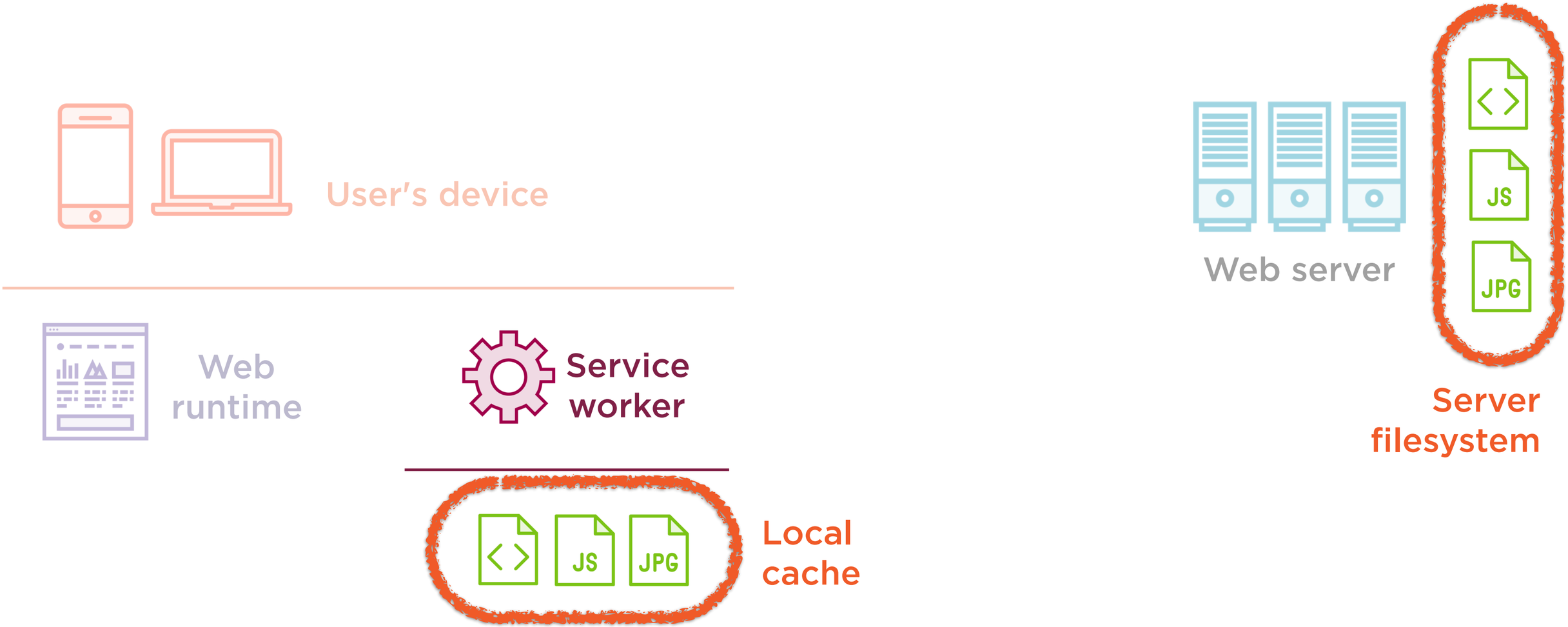
Caching and Serving PWA Resources

Progressive Web Apps

Progressive Web Apps



Progressive Web Apps



Caching Resources

- Service worker has a local cache
- We can cache all or some resources
- JavaScript promise
- Prefetch on installation
- Cache on request
- App Shell pattern

Serving Resources

The service worker will respond for every request the PWA make

It can serve from the cache

It can forward the request to the network

It can synthesize a response

Any mixed algorithm is possible

Cache Serving Strategies

Cache first

Network first

Stale while revalidate

Updating Resources

Files are saved in the client

Updating files in the server won't trigger any automatic change in the client

We need to define and code an update algorithm

It will need a process within your build system for hashing or versioning files

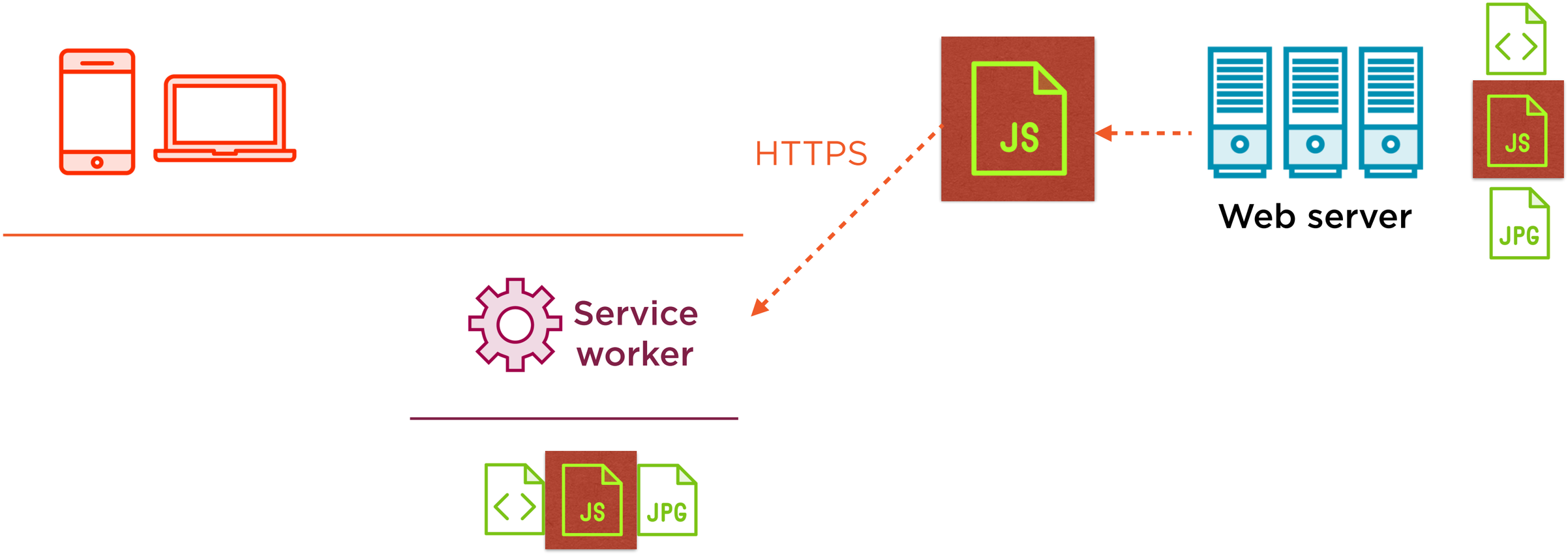
Developer is in full control of how to cache and serve the resources of the PWA, and how to manage API calls.

Updating the app doesn't require full reinstallation; we can just replace the updated files silently or with a user's notification.

Progressive Web Apps

Progressive Web Apps

Progressive Web Apps



Progressive Web Apps

Progressive Web Apps

Adding Service Workers to a Project

Middleware

A service worker is a middleware, so we can add it or remove it to any web project without architectural changes

Low level API

The service worker is a very low level API, with great power but also with not so many tools for common scenarios

PWA not mandatory

A service worker is mandatory for a PWA, but being a PWA is optional for using service worker's abilities

Adding a Service Worker

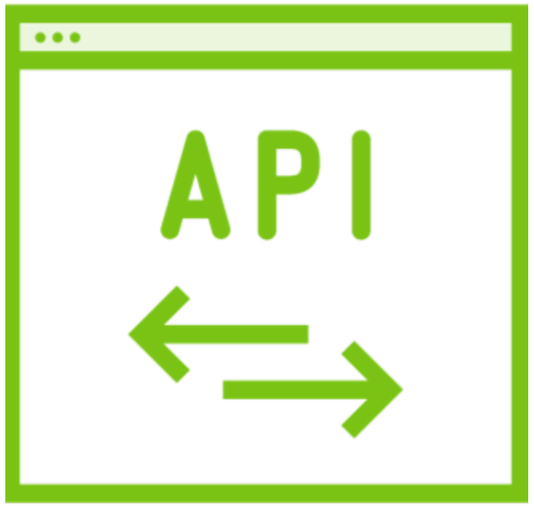
Define which abilities we'll use

Write the JavaScript file or use a library

Register it with the JavaScript API in your website or web app

Prepare your analytics and server

Define an update policy



Workbox JS

An open source library to manage the common design patterns for a service worker within a PWA

Workbox JS

- Managed by the Chrome team
- Precaching and runtime caching
- Request routing
- Updating resources
- Background sync
- Offline analytics
- Integration with Webpack, npm, Rollup
- Great flexibility with plugins

Doing PWAs with Angular

Angular CLI supports PWAs as a first-class output, part of the build system as an official add-on.

@angular/pwa

Using the CLI, you can add PWA support

It will create a Web App Manifest

It will generate a Service Worker, and a resources' manifest on each build

It can generate app shell for performance

It includes automatic update for PWA files

Routing setup is available through a JSON

Service Worker plugins are available

SwUpdate and SwPush services available

@angular/pwa

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG Search

Angular service worker introduction

Service workers augment the traditional web deployment model and empower applications to deliver a user experience with the reliability and performance on par with natively-installed code. Adding a service worker to an Angular application is one of the steps for turning an application into a Progressive Web App (also known as a PWA).

At its simplest, a service worker is a script that runs in the web browser and manages caching for an application.

Service workers function as a network proxy. They intercept all outgoing HTTP requests made by the application and can choose how to respond to them. For example, they can query a local cache and deliver a cached response if one is available. Proxying isn't limited to requests made through programmatic APIs, such as `fetch`; it also includes resources referenced in HTML, and even the initial request to

Doing PWAs with React

create-react-app supports PWAs as a first-class output, part of the build system.

PWAs with CRA

Using the CRA CLI, you have PWA support

It will create a Web App Manifest

It will generate a Service Worker, and a resources' manifest on each build

It includes automatic update for PWA files

It's disabled by default

It's not customizable, but it can be replaced by your own Service Worker logic

create-react-app and PWAs

Create React App

Docs Help GitHub Search

Making a Progressive Web App

EDIT

The production build has all the tools necessary to generate a first-class Progressive Web App, but **the offline/cache-first behavior is opt-in only**. By default, the build process will generate a service worker file, but it will not be registered, so it will not take control of your production web app.

In order to opt-in to the offline-first behavior, developers should look for the following in their `src/index.js` file:

```
// If you want your app to work offline and load faster, you c
// unregister() to register() below. Note this comes with some
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

As the comment states, switching `serviceWorker.unregister()` to `serviceWorker.register()` will opt you in to using the service worker.

PWAs Everywhere

PWAs Everywhere

Many CLIs and build systems now create Progressive Web Apps

Most are based on Workbox

Plugins for Wordpress, Drupal and CMS

PWA builders and starter kits

Vue CLI, Ionic, Salesforce, Polymer

Summary

Programming the PWA with web tools

- Service Workers: the brain of a PWA
- Caching and serving resources
- Adding a Service Worker
- Doing PWAs with Angular
- Doing PWAs with React
- PWAs everywhere

Next up: Distributing the app to your users