Progressive Web Apps (PWA) with Angular

Rocking the web the native way!

think tecture

Steffen Jahr
@steffenjahr

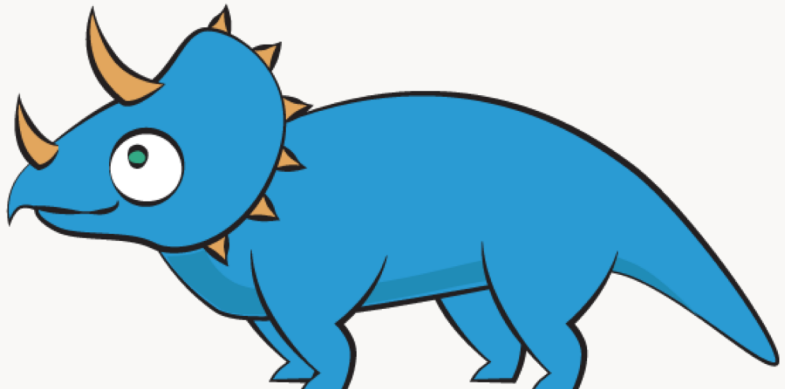
Christian Liebel
@christianliebel

Things TO EXPECT

- Extensive/up-to-date insights into PWA and Angular's PWA support
- Hands-on labs for you
- A PWA use case that works on your (comparably modern) Android phone and Chrome/Firefox browsers
- Latest & greatest software
- Lots of fun

Things **NOT** to expect

- Blueprint for PWA development
- Future-proof tutorial for PWA development with Angular (experimental support)
- Extensive one-on-one support



Your instructors

Feel free to ask questions anytime!

Steffen Jahr

Christian Liebel

Timetable

09:30 – 11:00	**Block 1**
11:00 – 11:20	Break
11:20 – 12:45	**Block 2**
12:45 – 14:00	Lunch Break
14:00 – 15:30	**Block 3**
15:30 – 16:15	Break
16:15 – 17:30	**Block 4**

Intro

- Motivation
- Cordova/Electron
- "Uber Pattern"

Web App Manifest

- Add to Home Screen
- App Install Banners

Service Worker

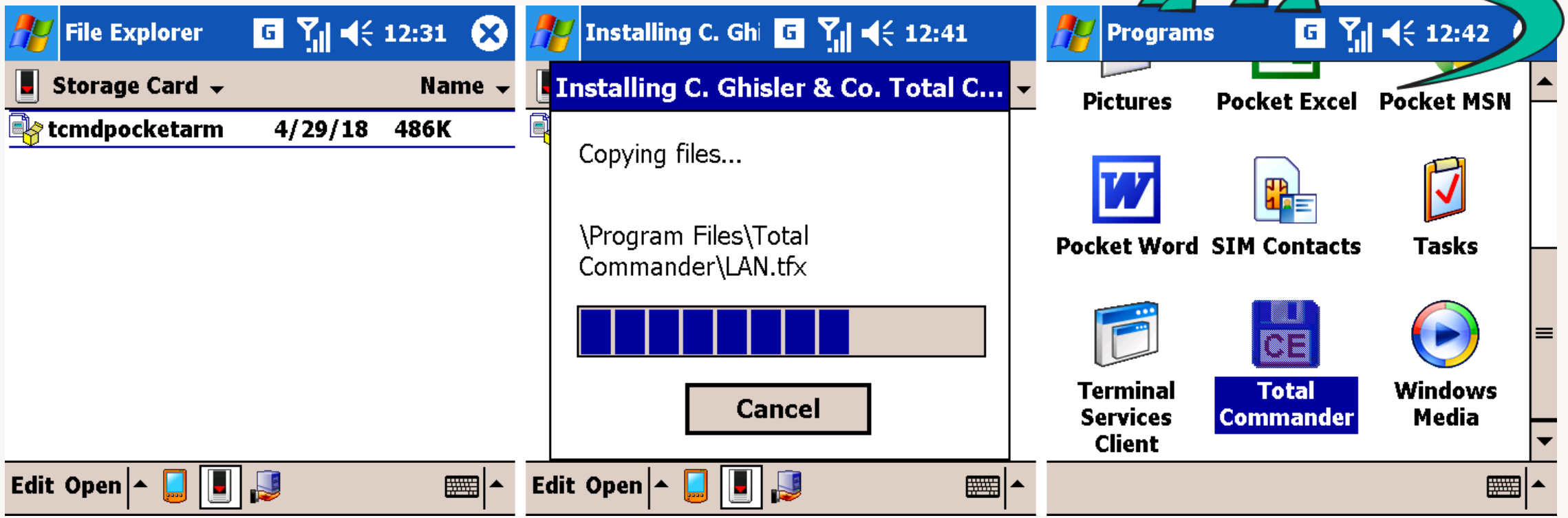
- Lifecycle
- App Shell
- Caching
- Offline-First & Sync

Advanced

- Push Messages
- Tools
- Limitations

Apps in 2005...

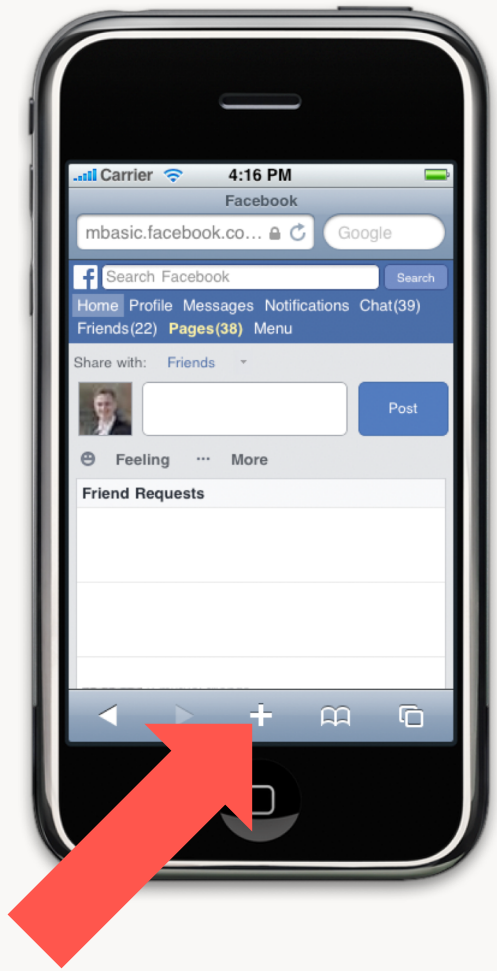
Windows Mobile 5



Apps in 2007...

https://www.youtube.com/watch?v=vN4U5FqrOdQ

> And guess what? There's no SDK that you need! You've got everything you need if you know how to write apps using the most modern web standards to write amazing apps for the iPhone today.
>
> — Steve Jobs, June 11, 2007


Carrier
4:16 PM
Facebook
mbasic.facebook.co…
Google
Search Facebook
Search
Home Profile Messages Notifications Chat(39)
Friends(22) Pages(38) Menu
Share with: Friends
Post
Feeling ··· More
Friend Requests

> Let me just say it: We want native third party applications on the iPhone, and we plan to have an SDK in developers' hands in February.
>
> — Steve Jobs, October 17, 2007

Real Cross Platform – Get It Today

Native App Packaging

APACHE CORDOVA™

ELECTRON

Real Cross Platform – Get It Today

Native App Packaging

APACHE CORDOVA™

Native web view
(WKWebView/UIWebView,
Android Browser/Chrome, Edge)

+

Native plug-ins
(Objective-C, Java, C#, ...)

ELECTRON

Node.js runtime

+

Chromium engine

Real Cross Platform – Get It Today

Native App Packaging

APACHE CORDOVA™

ELECTRON

Result: App packages

.ipa, .apk, .appx



Google Play

Result: Executables

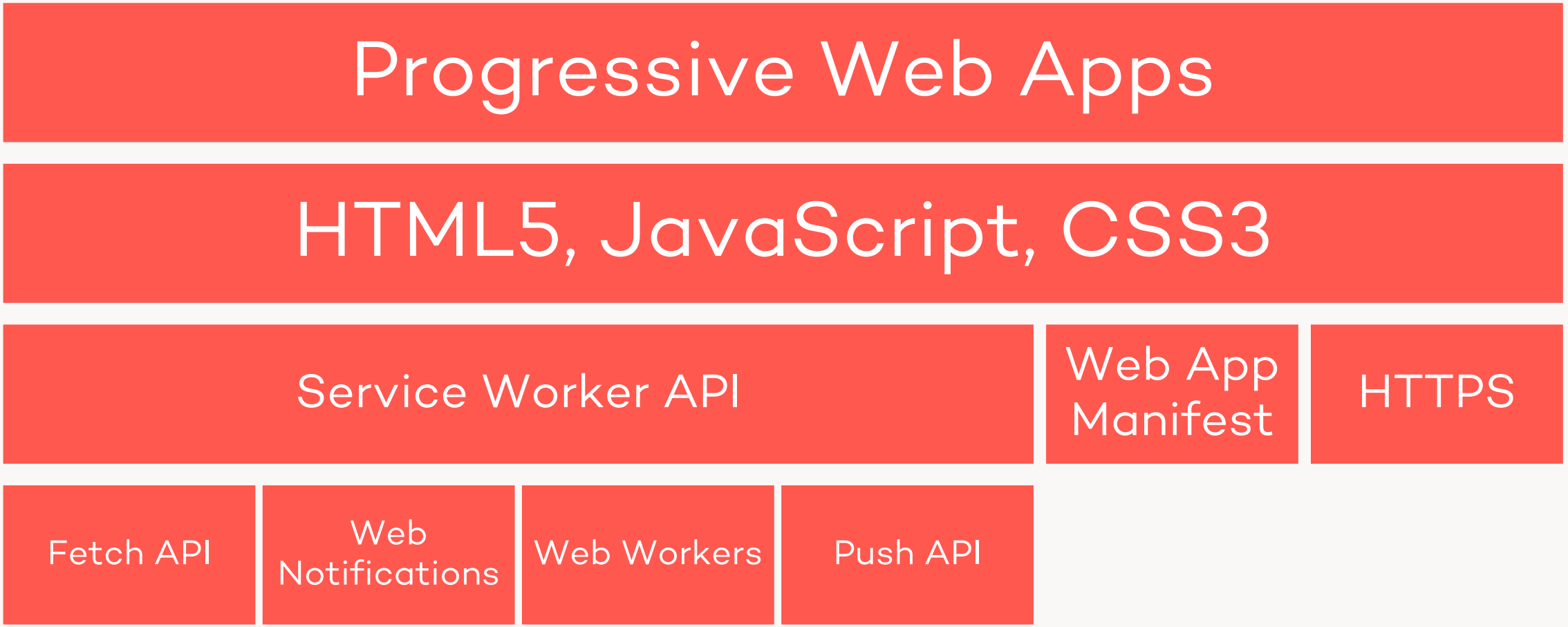
.exe, .app, ELF

Apps Tomorrow...

Web Goes Native

- No app stores anymore!
- Web App = App App
- Feature parity: Push notifications, offline availability and more for web & native applications
- Backwards compatibility: PWAs can also be run on non-PWA browsers, but with reduced feature set

PWA Technology Overview



The Power of the Modern Web

WebRTC

Media Capture and Streams

Geolocation

Web Notifications

Payment Request API

Web Share API

WebGL

Generic Sensor API

Gamepad API

Speech Synthesis

Canvas

Web Bluetooth API

Shape Detection API

IndexedDB

WebVR

Web Cryptography API

The Power of the Modern Web

Flight Arcade

- WebGL
- Web Audio API
- Gamepad API

DeviantArt | DanRabbit

PWA Status Quo

Platform Support

40

44

11.1

17

4.1

Chrome 40

11.3

PWA Status Quo

This looks great, but there are still…

features

- that aren't exposed to the web platform (e.g. Contacts API)
- that aren't available on every platform (e.g. Ambient Light API)
- that require more performance (e.g. PDF annotations)

devices

- that don't support Progressive Web Apps (very old Android devices, old iOS devices, etc.)

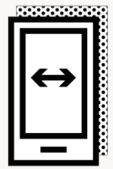
marketing reasons

- App Store presence

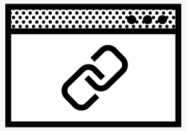
“Uber Pattern”

Progressive Web Apps are
not a technology,
but a **collection of features**
an application must/should support.

“Uber Pattern”



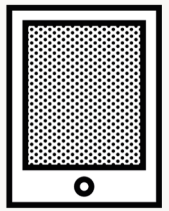
Responsive



Linkable

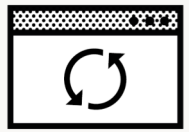
Discoverable

Installable



App-like

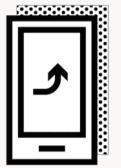
Connectivity Independent



Fresh

Safe

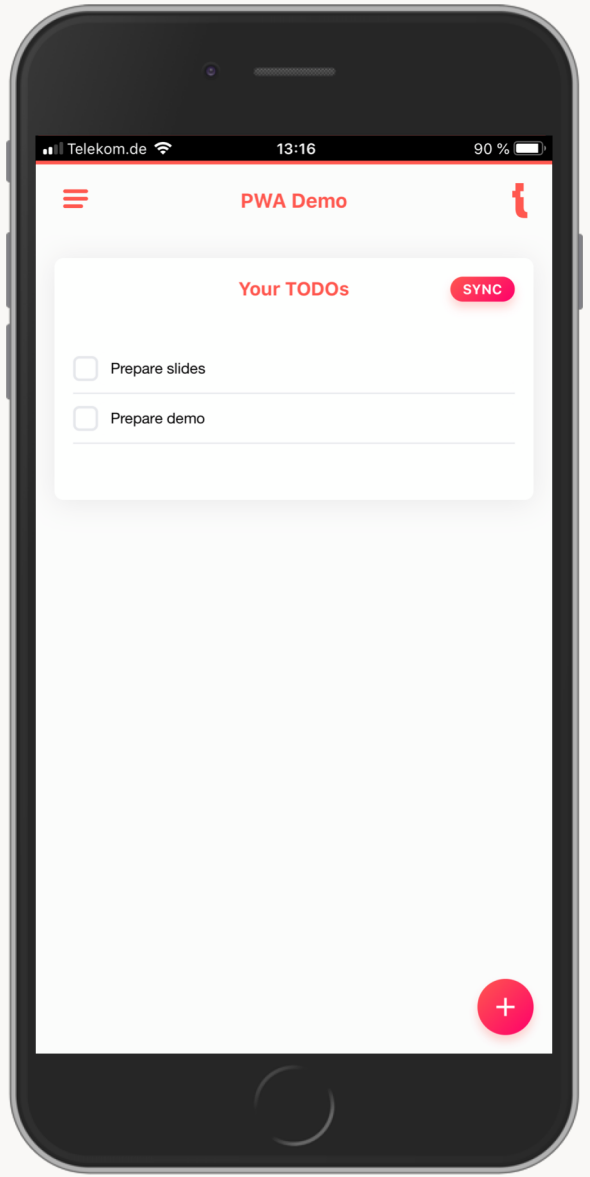
Re-engageable



Progressive

Demo Use Case

- The next best **to-do list app**
- Create, read, update and delete to-dos
- Progressive Web App: installable
- Responsive: one size fits all
- Offline-first: synchronize to-do list with backend using IndexedDB & Dexie.js
- Web Share API: Share your to-dos with just a tap
- No user context, sorry



Demo Use Case

Live Demo

Let's get started

Setup complete?

Let's get started

Repository:
https://bit.ly/ac-pwa

LAB #0

What about Angular & PWA?

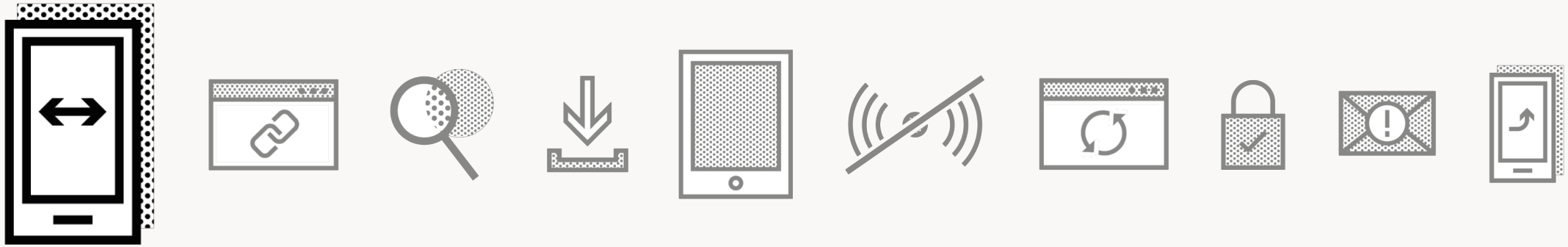
Angular CLI & PWA Schematic

Existing Projects

```
ng add @angular/pwa
```

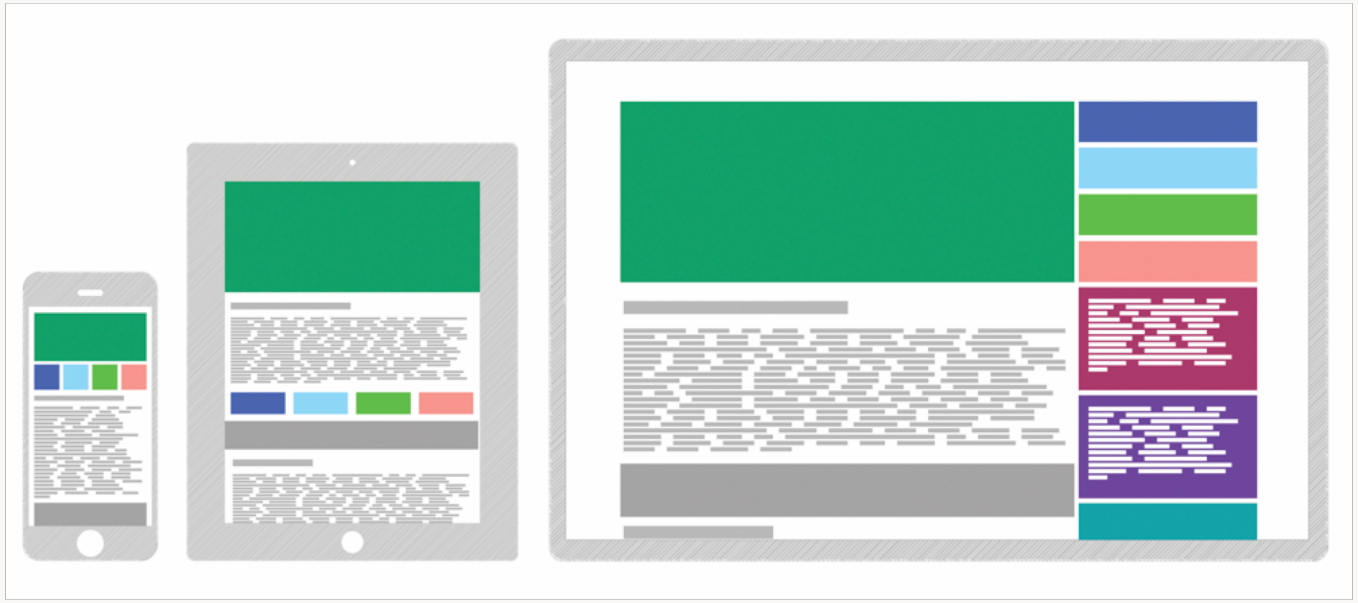
LAB #1

PWA Features



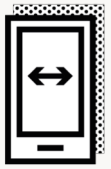
Responsive

Responsive



Responsive

One size fits all



A **single website layout** for different screen resolutions (typically related to the screen's width)

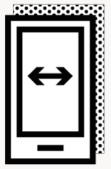
Definition of **trigger points** at which the layout changes

Mobile first: Create (minimal) mobile views first, then proceed with larger screens, so all devices get the best experience

Key technologies: CSS3 Media Queries, CSS Flexible Box Layout, CSS Grid Layout

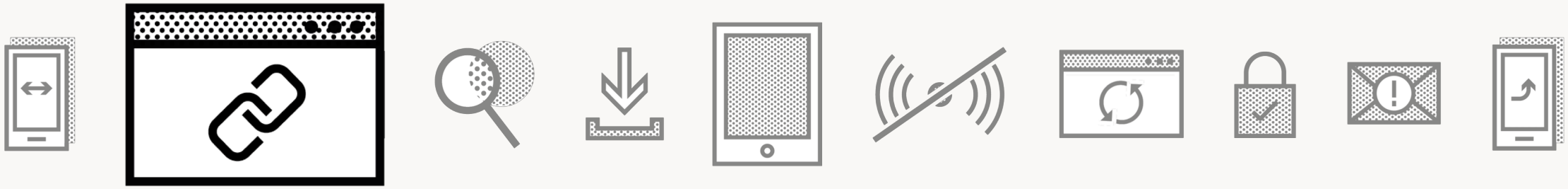
Responsive

Frameworks



- Angular Material
- ngx-admin
- Bootstrap
- Foundation
- Custom-tailored solution

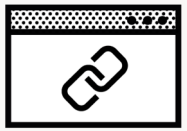
PWA Features



Linkable

Linkable

Reference Apps and Internal App States



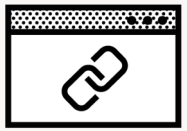
Goal: Directly link to applications or application states/views (deep link)

Key technology for references in the World Wide Web: Hyperlinks

Schema for hyperlinks: **Uniform Resource Locator** (URL)

Linkable

Reference Apps and Internal App States



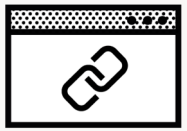
Approximation in (mobile) operating systems: URI schemes (e.g. `fb://profile`)

For **applications** on the web, this comes for free

In some cases also for **application states/views** (e.g. for single-page applications)

Linkable

Reference Apps and Internal App States



Classic (Query String)

https://myapp.com/profile.php?id=peter.mueller

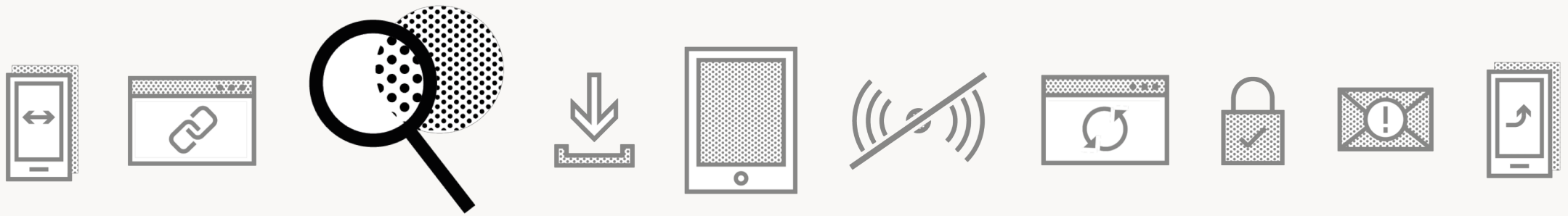
Hash Anchor (e.g. SPA)

https://myapp.com/#/profile/peter.mueller

HTML5 History (SPA)/Server Rendered

https://myapp.com/profile/peter.mueller

PWA Features



Discoverable

Discoverable

Distinguish Web Apps from Websites

How to distinguish websites from apps?

Idea: Add a file with metadata for apps only

Advantage: Apps can be identified by search engines, app store providers etc.

Web App Manifest

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

Web App Manifest

manifest.json

```
{
  "short_name": "PWA Demo",
  "name": "AngularConnect PWA Demo",
  "icons": [
    {
      "src": "assets/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone"
}
```

Title

Icons

Start URL

Display Type

Related Apps

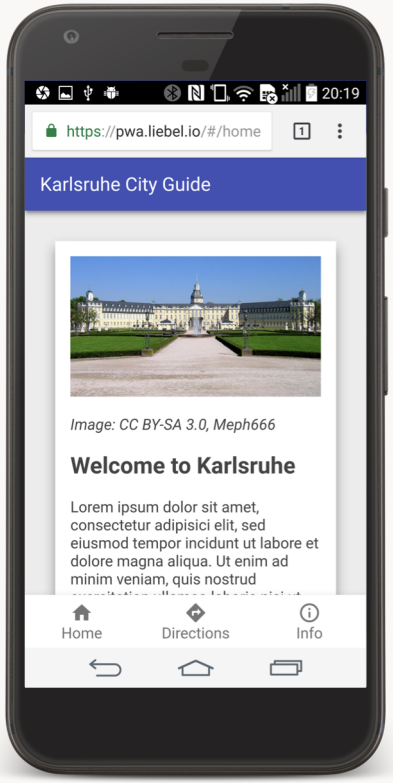
Description

Age Rating

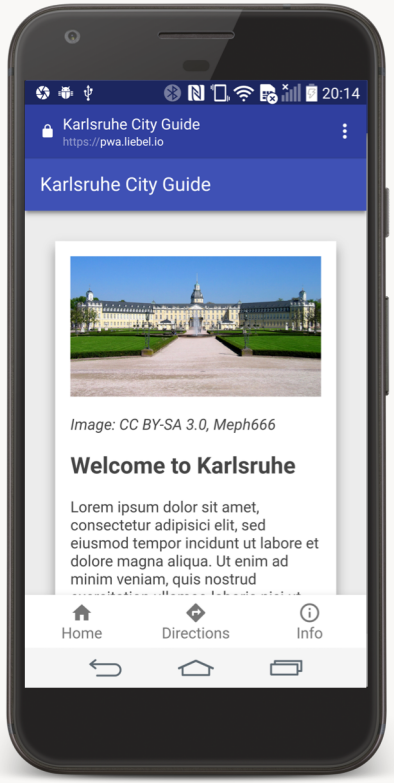
Additional Config

Web App Manifest

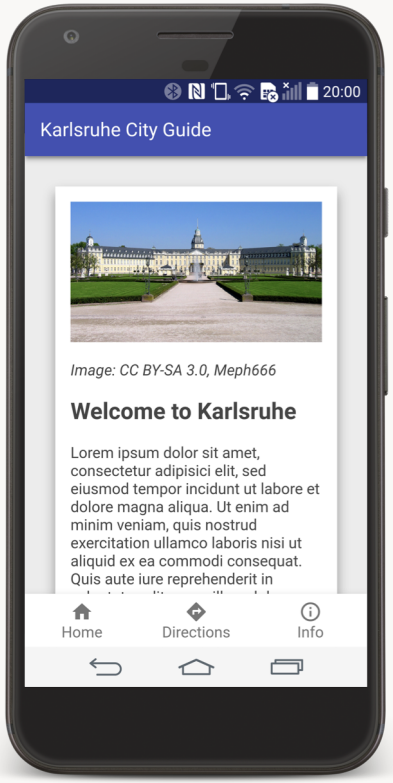
Display Modes



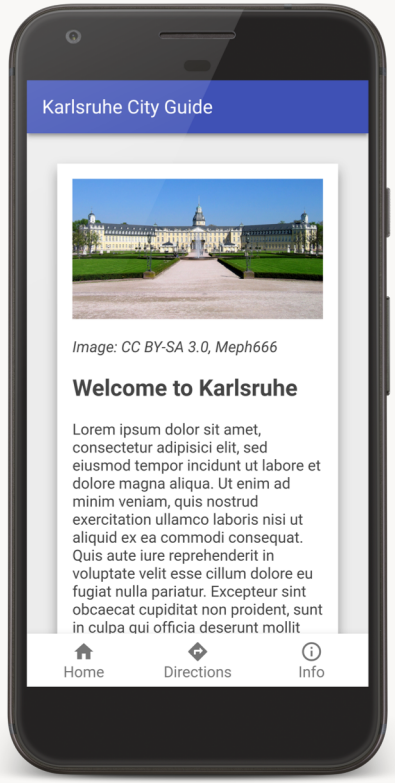
browser



minimal-ui



standalone



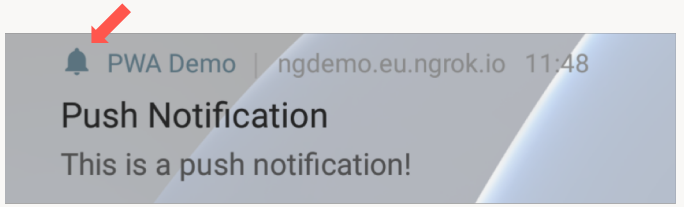
fullscreen

Web App Manifest

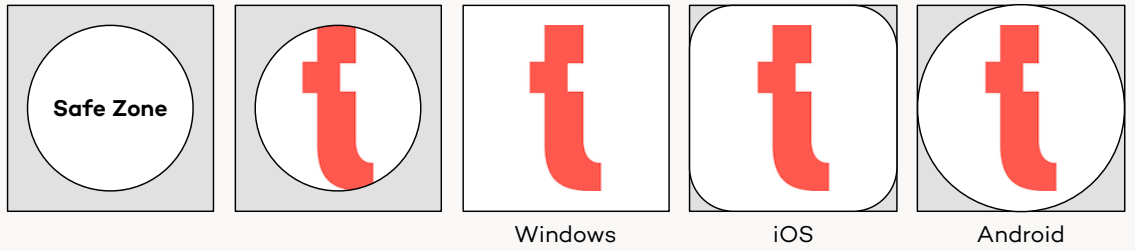
Icon Purposes

(any)
any context (e.g. app icon)

badge
different space constraints/
color requirements



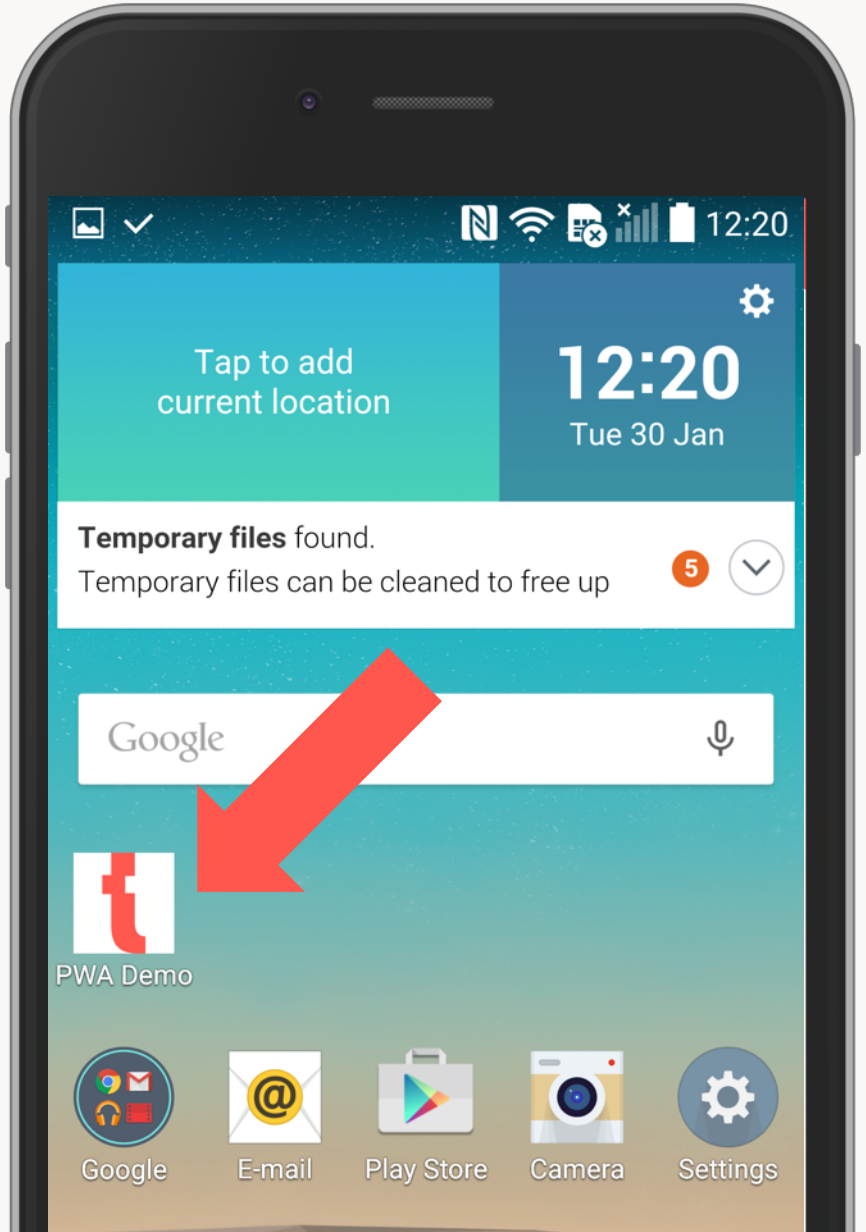
maskable
user agent masks icon as required



Web App Manifest

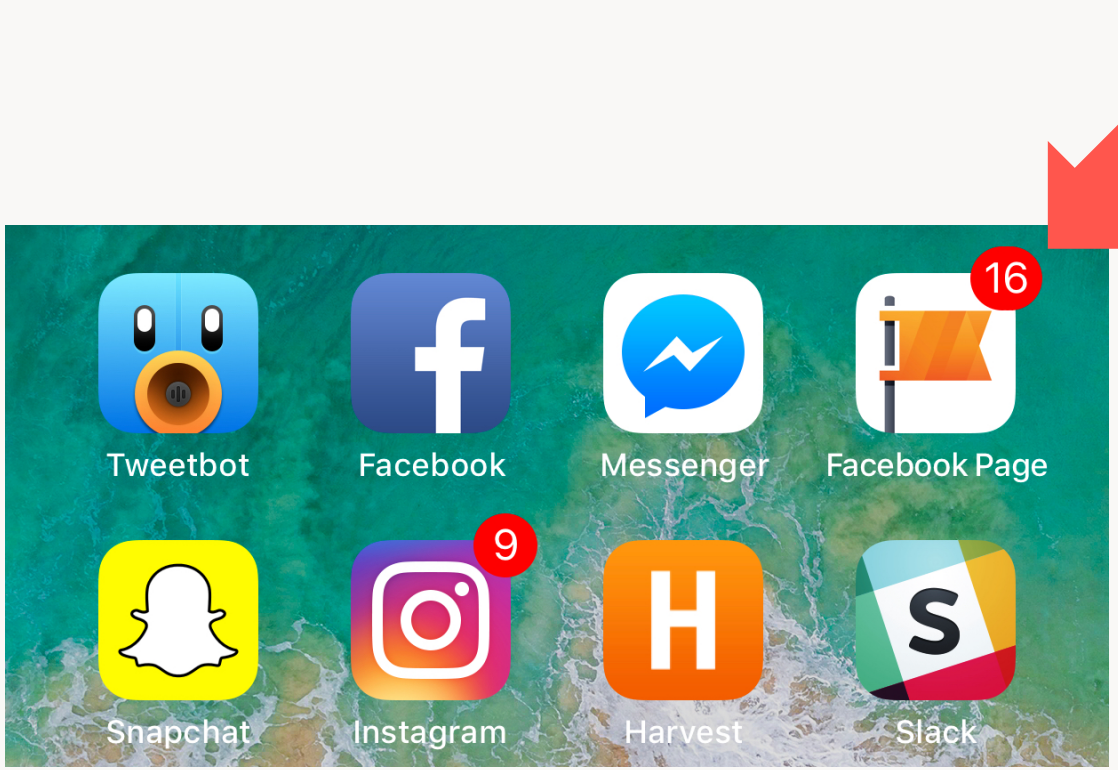
manifest.json

```
{
  "short_name": "PWA Demo",
  "name": "AngularConnect PWA Demo",
  "icons": [
    {
      "src": "assets/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone"
}
```



Web App Manifest

Badging API



PWAs & Windows Store

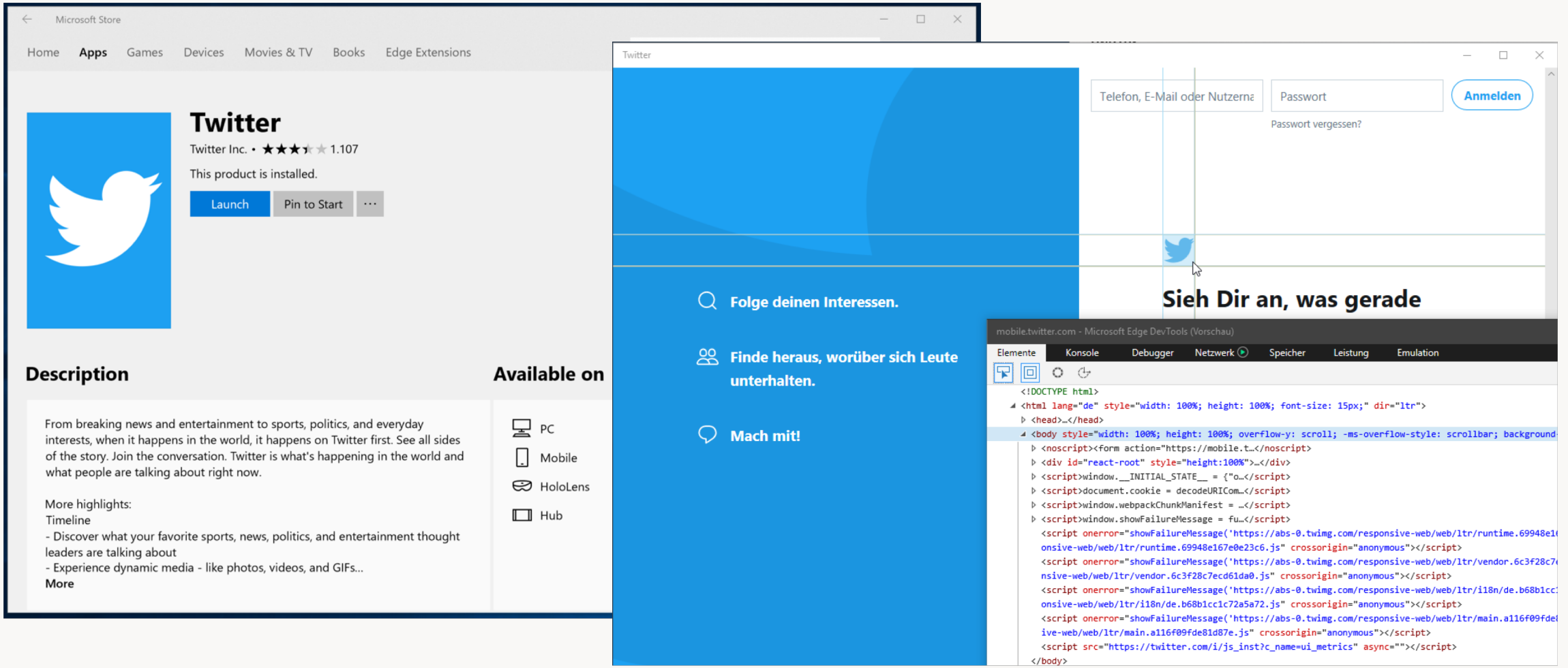
Auto indexing (beta)

High-quality PWAs are automatically discovered, reviewed and published in the store by Bing

Self publishing

Developer publishes the App in the Microsoft store under his own name

PWA & Windows Store

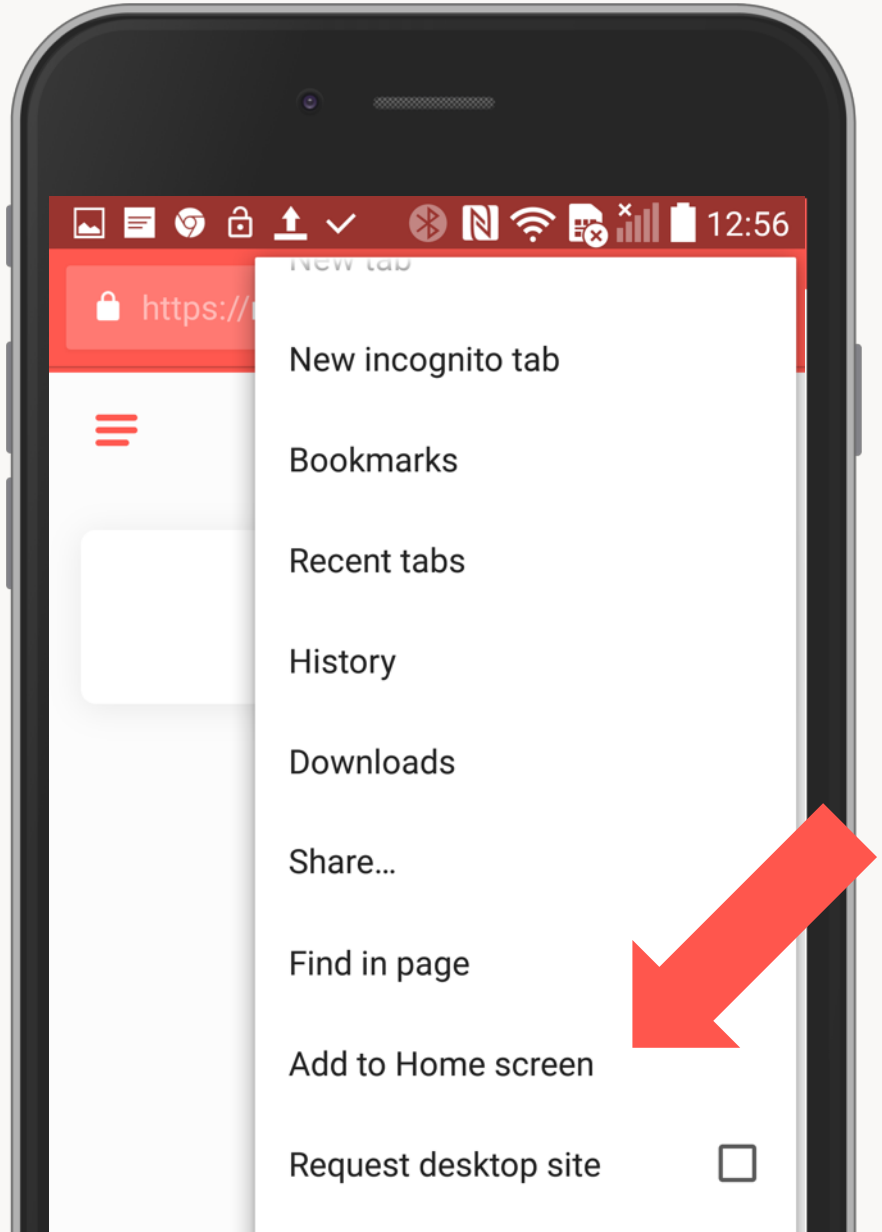


Web App Manifest

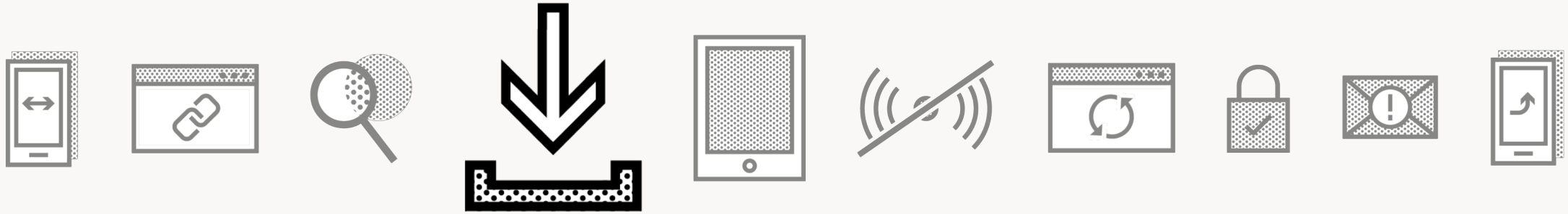
Adjust the Web App Manifest to your needs

Test it by adding the app to your home screen

LAB #2



PWA Features

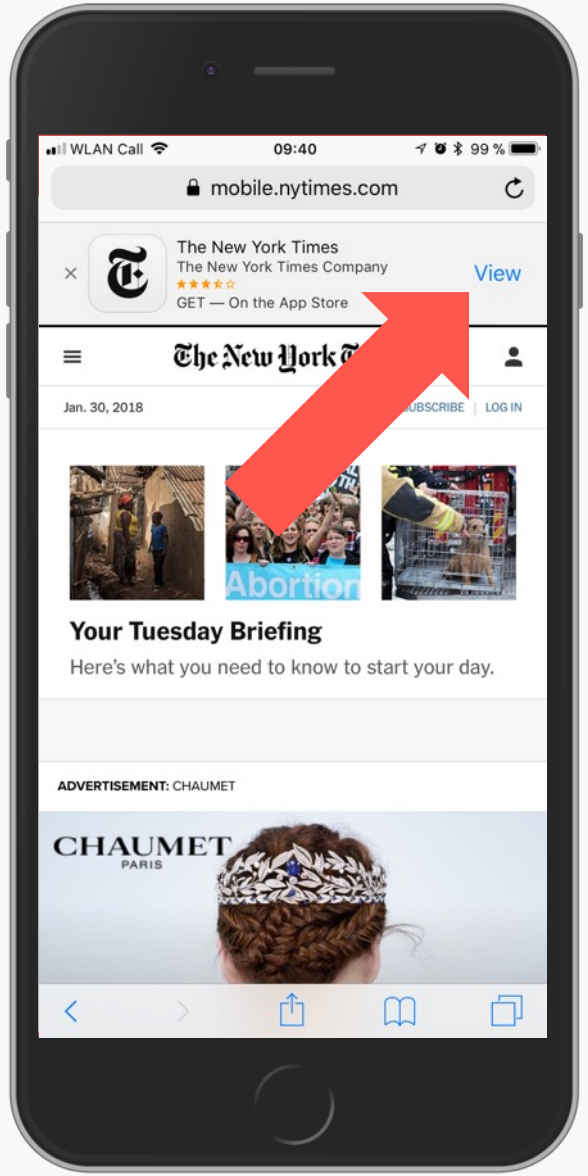


Installable

Installable

App Install Banners

Website suggests installing a related native app (e.g. known from iOS, link to App Store)



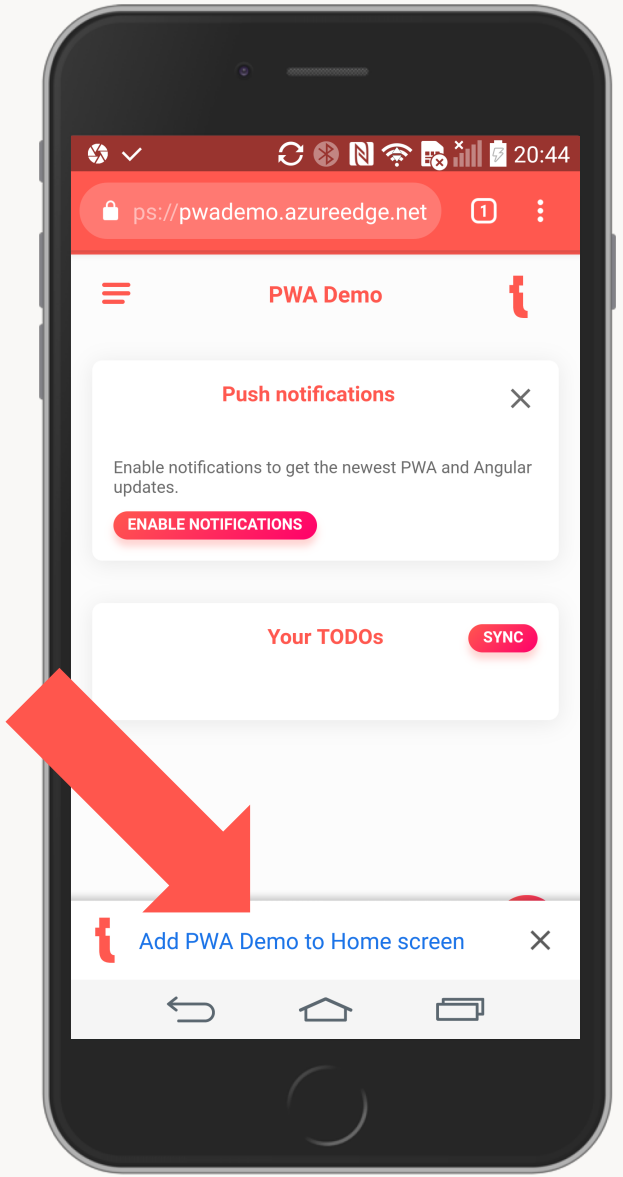
Installable

App Install Banners

PWA: Web app and installed app are one and the same, so the web app can be directly added to the homescreen instead

You've seen this before: iOS Add to homescreen, IE9 Pinned Websites, Windows 8 Live Tiles etc.

If certain criteria are met, Google Chrome will prompt the user to add the PWA to the homescreen (for a transitional period)



Installable

App Install Banners

For the web as a part of the **Web App Install Enhancement Proposal**

Shows metadata defined in the **Web App Manifest** file

Browser controls when the minibar is shown based on certain criteria

Event: `beforeInstallPrompt`

Displaying the minibar can be prevented/deferred by the application

Application can determine if minibar was dismissed/confirmed

Installable

Prevent/Defer App Install Banners

```
let deferredPrompt;

window.addEventListener(
'beforeinstallprompt', e => {
      // for Chrome <= 67
      e.preventDefault();
      deferredPrompt = e;
});
```

```
btnClick.addEventListener(
'click', () => {
  if (deferredPrompt) {
    deferredPrompt.prompt();
  }
});
```

Installable

App Install Banners – Criteria for Google Chrome

Web App is not already installed

Meets a user engagement heuristic (currently: user has interacted with domain for at least 30 seconds)

Includes a Web App Manifest that has `short_name` or `name`, at least a 192px and 512px icon, a `start_url` and a `display` mode of `fullscreen`, `standalone` or `minimal-ui`

Served over HTTPS

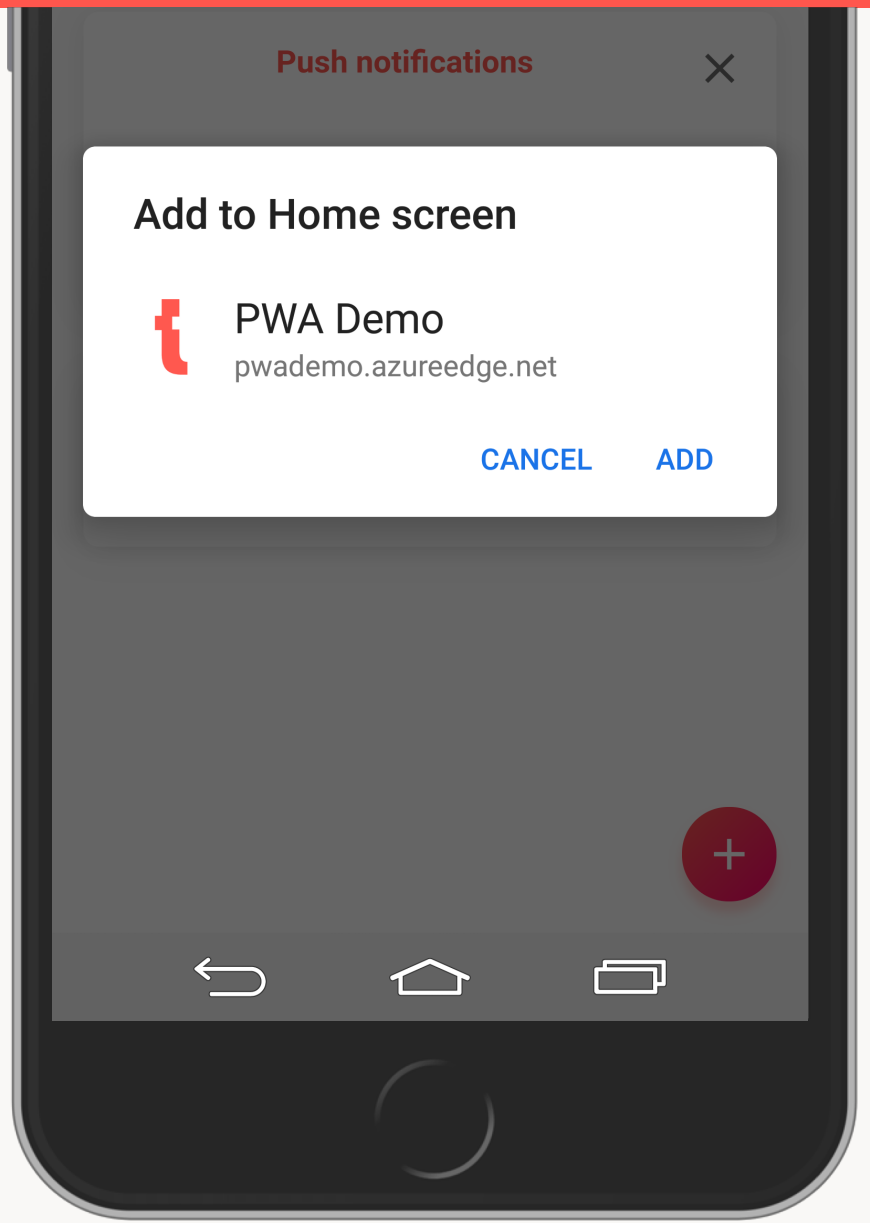
Has a registered service worker with a `fetch` event handler

https://developers.google.com/web/fundamentals/app-install-banners/

App Install Banner

manifest.json

```
{
  "short_name": "PWA Demo",
  "name": "Thinktecture PWA Demo",
  "icons": [
    {
      "src": "assets/icon-144x144.png",
      "sizes": "144x144",
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone"
}
```

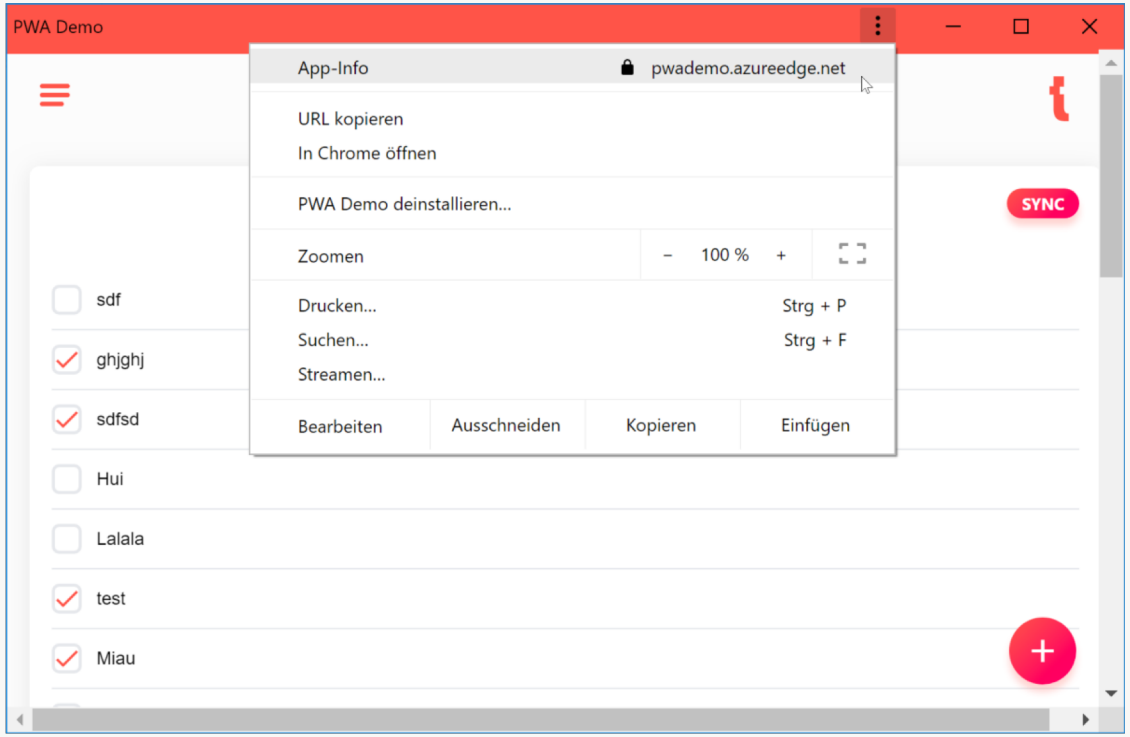


Installable

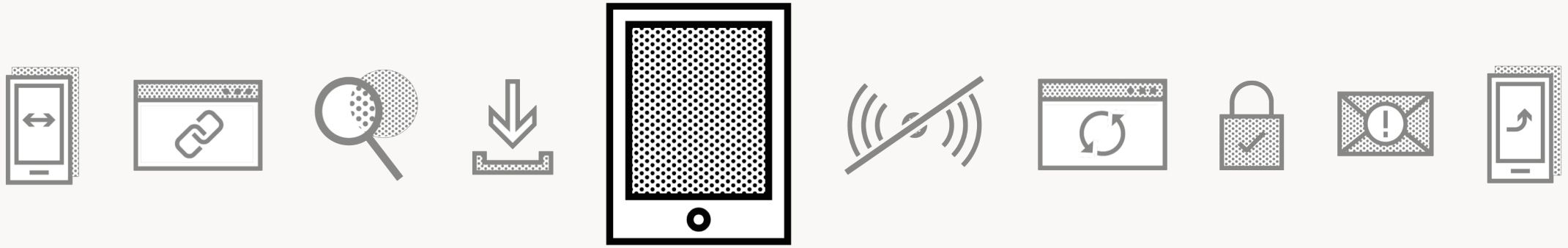
And linkable...?

Desktop PWAs show their origin in the title bar (on Windows)

Link capturing is on Google's roadmap, so an installed PWA opens when clicking a link of a certain origin instead of the browser



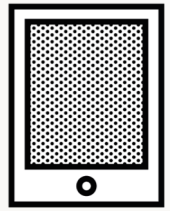
PWA Features



App-Like

App-like

Web Apps that feel natively

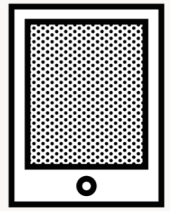


Native functionality with HTML5 technologies

- Playing audio and video
- Hardware accelerated 2D/3D graphics
- Gamepad
- Touch input
- Local storage
- Location-based services
- Access to camera
- etc.

App-like

Single-Page Web Applications



Pattern how to develop apps in a native style: Single-Page Web Applications (SPA)

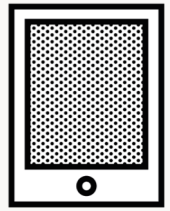
Views, logic and assets are stored locally (e.g. in browser cache)

App navigation doesn't lead to a complete page reload (server round-trip)

SPAs only request server for getting or modifying data via APIs

App-like

Single-Page web applications



Approach: **Mobile First**

Start with mobile devices when designing use cases and user interfaces

Extend for devices with larger screens and more precise input methods at a later point of time

App-like

App Shell

Certain parts of the UI are always visible

Other parts are dynamic content

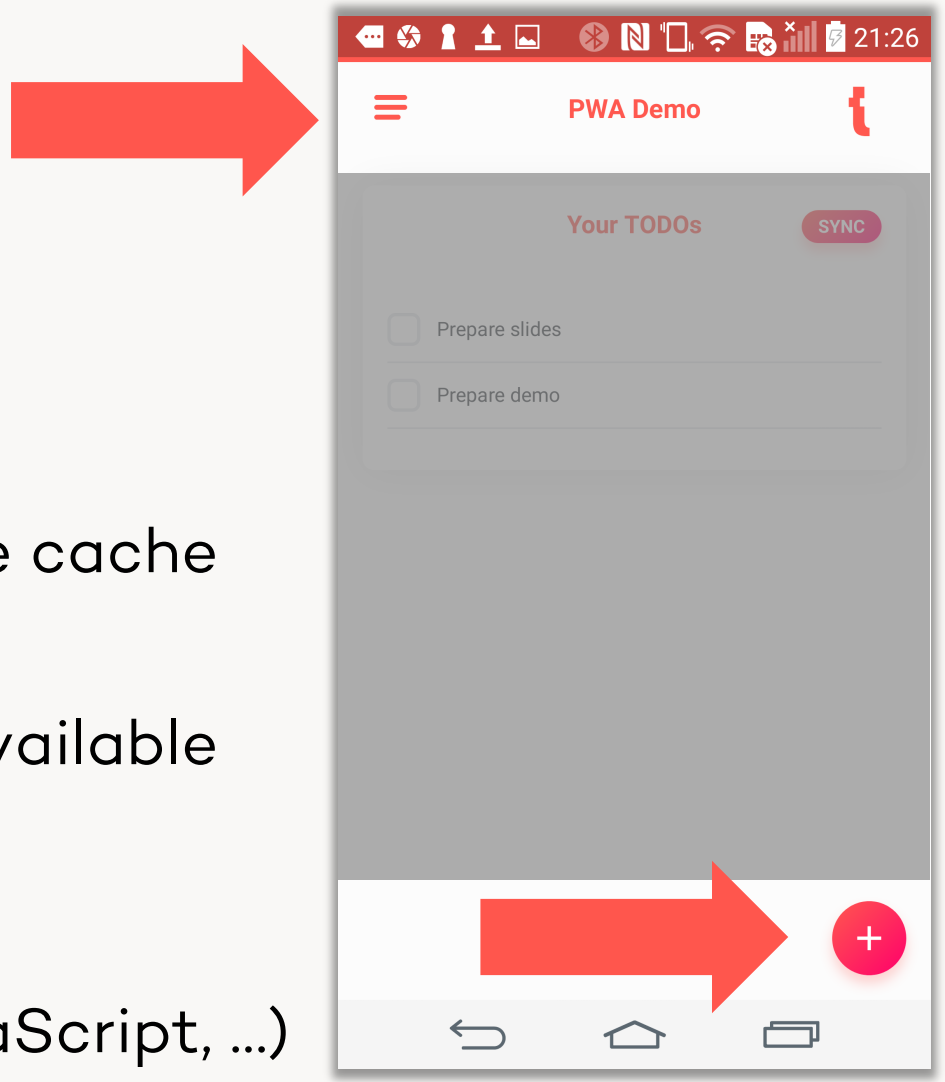
Idea: Static parts (App Shell) are stored in the cache

App Shell works without or bad connectivity

Example: Error message if no connection is available

Technology of the App Shell doesn't matter

(Angular, AngularJS, React, jQuery, plain JavaScript, ...)



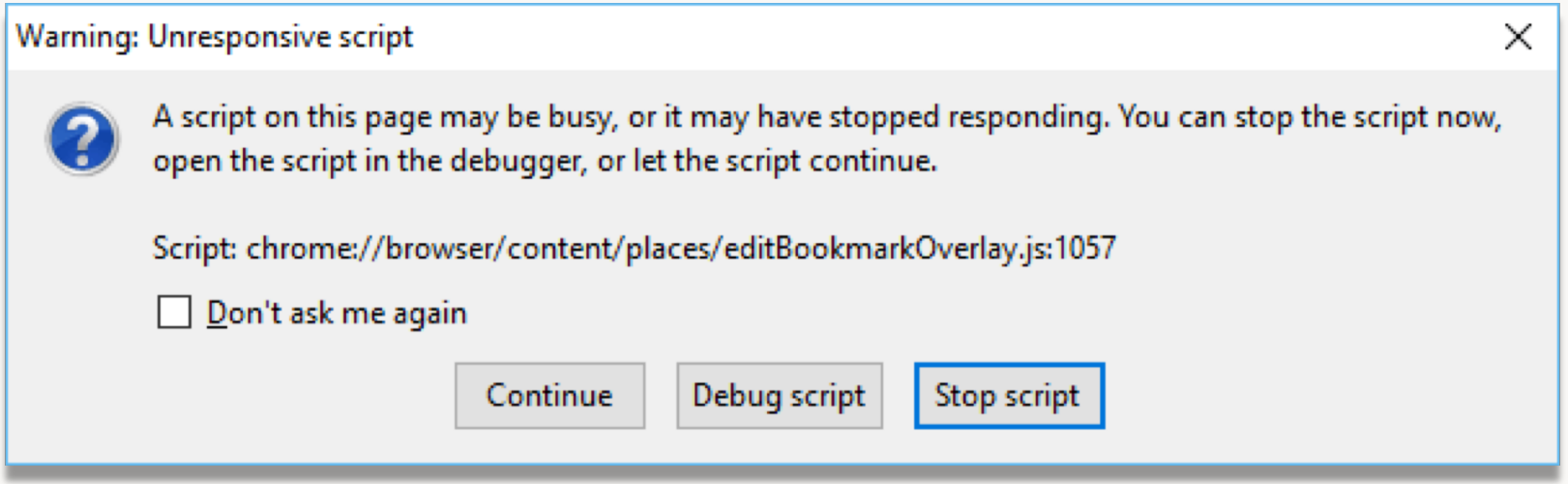
PWA Features

Connectivity Independent

Part 1

Web Worker
Shared Worker
Service Worker

JavaScript & Threading



http://techdows.com/wp-content/uploads/2016/01/Firefox-unresponsive-JavaScritp-edit-bookmarkOverlay.png

Web Worker

```
var worker = new Worker('script.js');
```

Worker snippet is executed in an own thread

Worker can't manipulate the parent document's DOM

Communication via thin API (postMessage)

Relation: Current tab/window

Lifetime: Until tab/window is closed

Shared Worker

```
var worker = new SharedWorker('script.js');
```

Worker snippet is executed in an own thread

Worker can't manipulate the parent document's DOM

Communication via thin API (postMessage)

Relation: Origin (protocol + hostname + port)

Lifetime: Until all tabs/windows of an origin are closed

Service Worker

Worker snippet is executed in an own thread

Worker can't manipulate the parent document's DOM

Communication via thin API (postMessage)

\+ Acts as a controller/proxy/interceptor

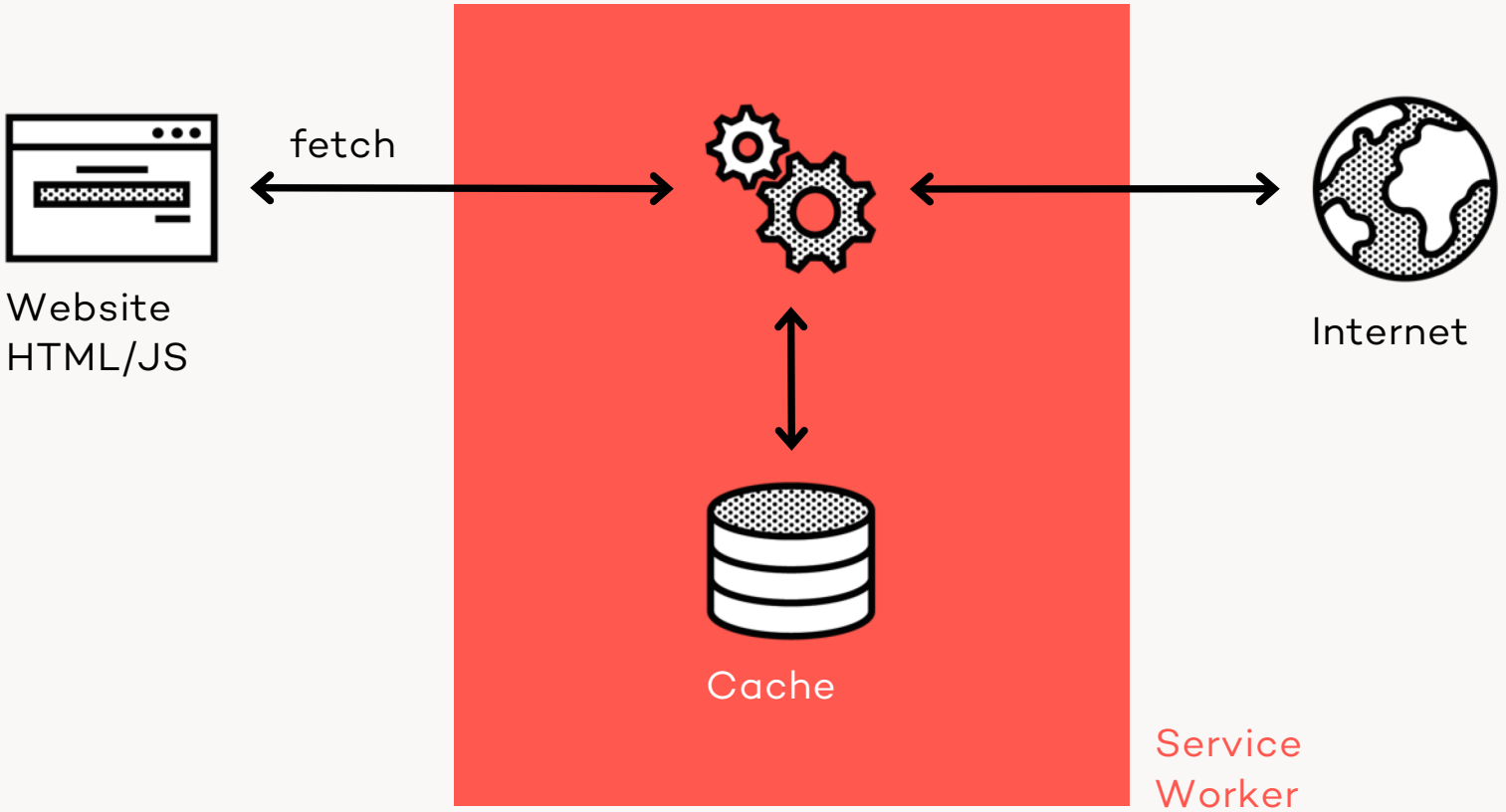
\+ Performs background tasks

Has to be installed before usage

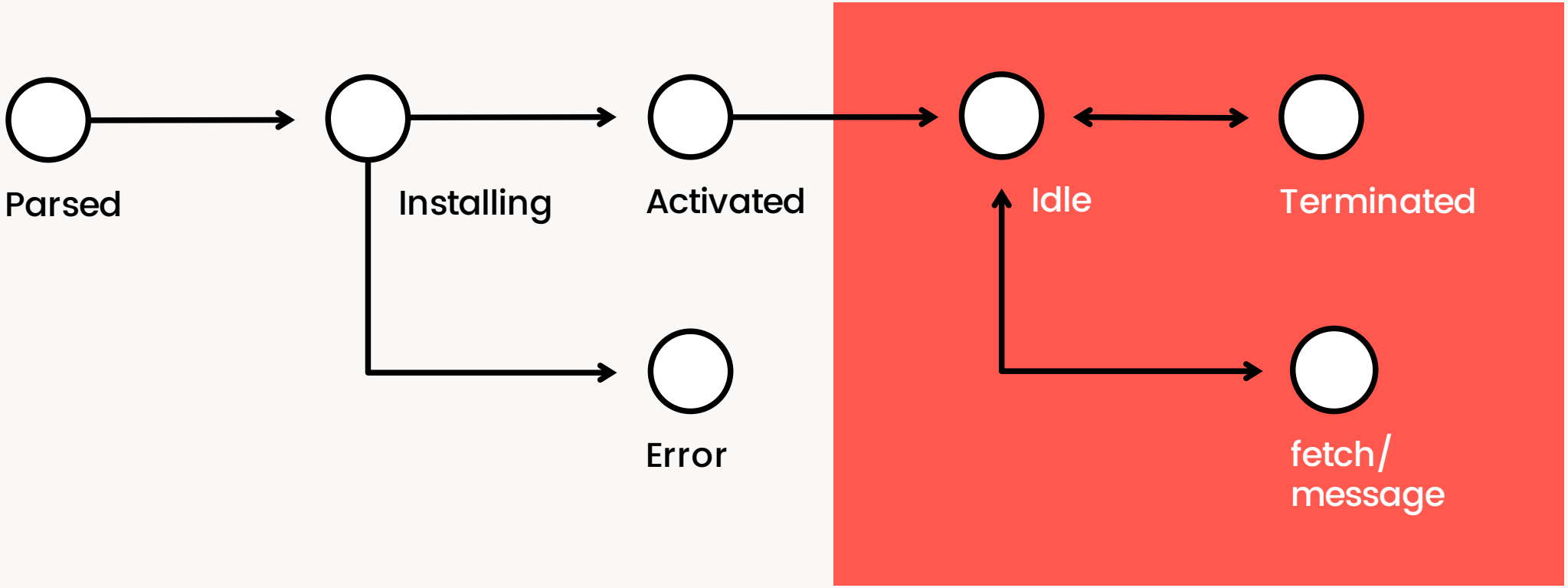
Relation: Scope (origin + path)

Lifetime: Unrelated to tab/window

Service Worker as a controller/proxy/interceptor



Service Worker Lifecycle



Connectivity Independent

Offline capability

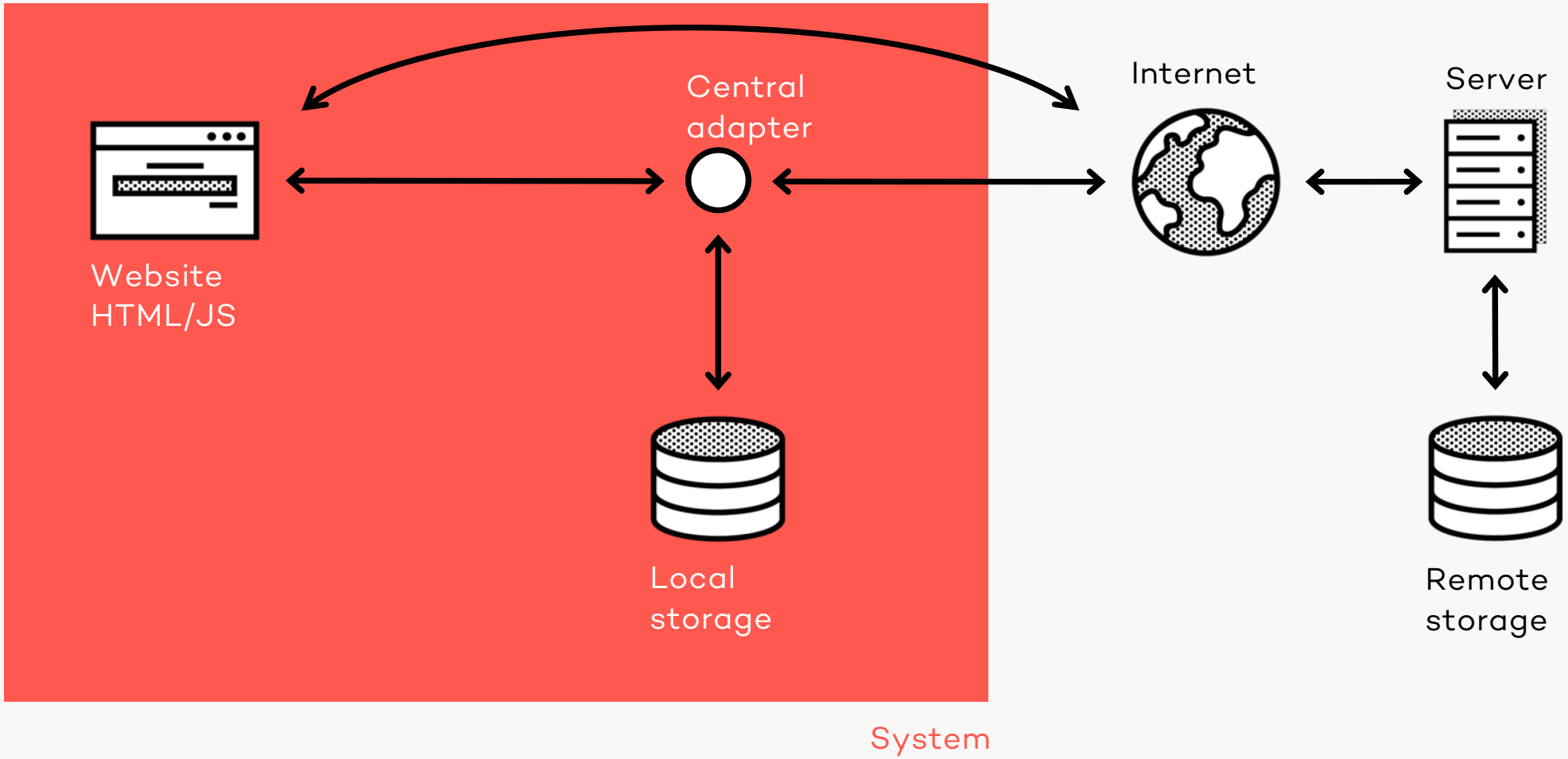
Challenge: Connection strength varies a lot (especially en route)

Lie-Fi: Connection strength of a public WiFi is weak or even completely offline

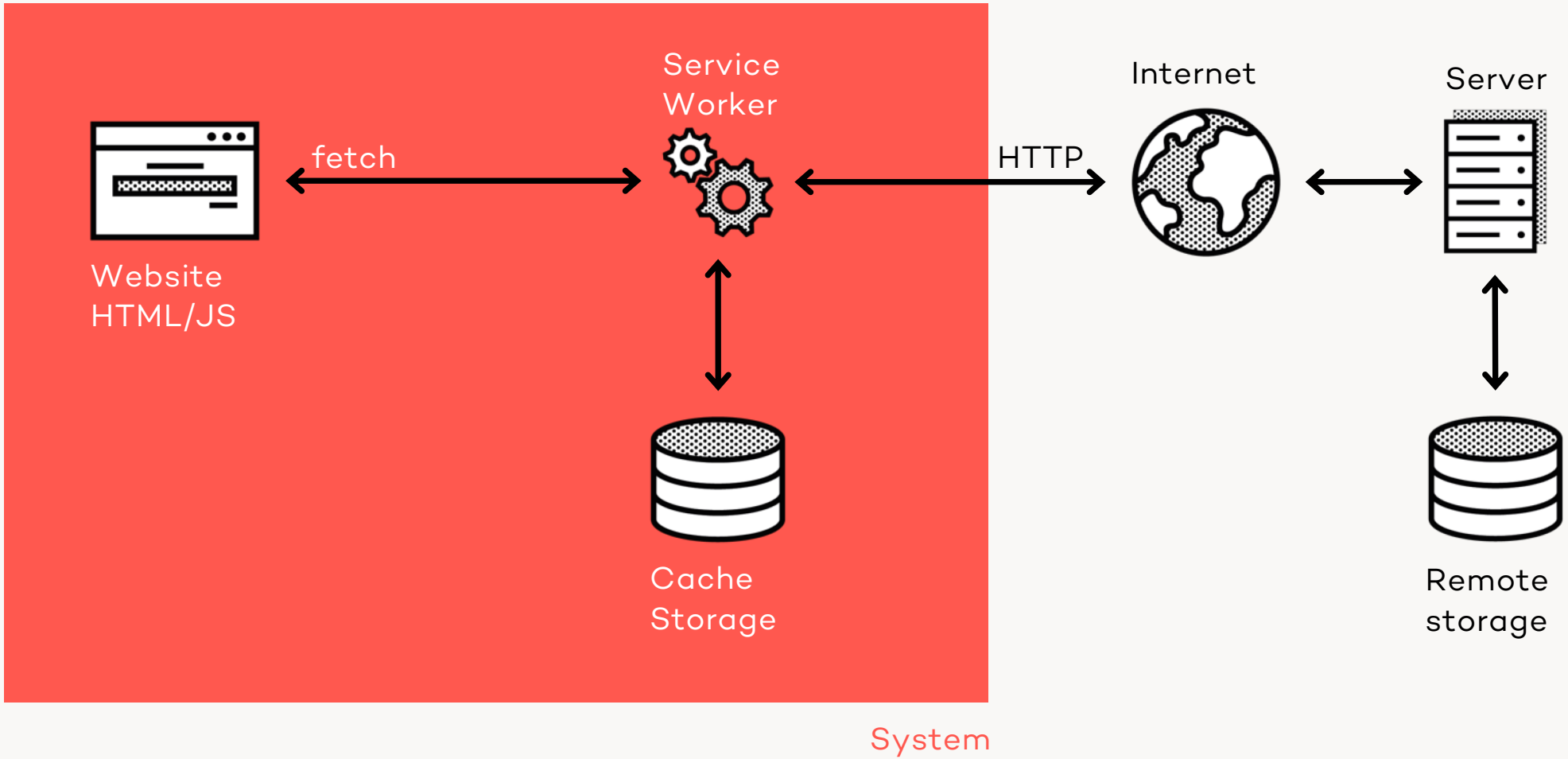
Goal: App works offline or with a weak connection at least within the possibilities (e.g. OneNote)

Hence: Local data storage is needed—synchronization/conflict management required (Web Background Synchronization)

Offline capability in General



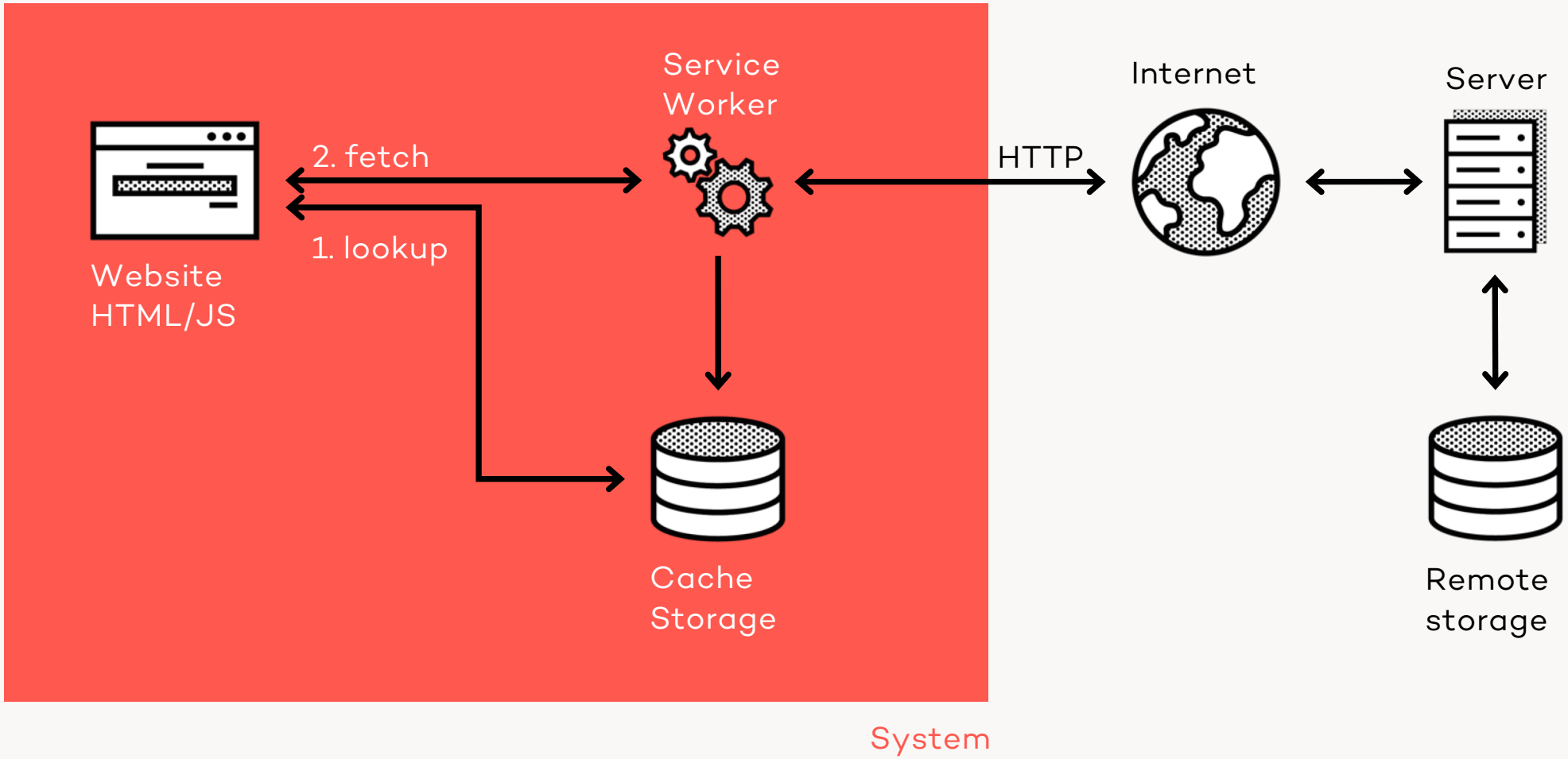
Offline capability with Service Worker



Caching Strategies

1. Cache only
2. Network only
3. Cache falling back to network
4. Cache & network race
5. Network falling back to cache
6. Cache then network
7. Generic fallback
8. ServiceWorker-side templating

Cache Then Network



@angular/service-worker

Overview

Implements a “one-size-fits-all” service worker

Instrumented via `ngsw-config.json`

Restricted feature set (e.g., no communication between app and SW)

- Initial caching of static content
- Caching external content
- Dynamic Caching
- Push notifications

@angular/service-worker

ngsw-config.json

```
{
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/*.js",
        ]
      }
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
        ],
        "urls": [
          "http://xy.invalid/1.jpg"
        ],
      }
    }
  ]
}
```

@angular/service-worker

CLI Integration

Angular CLI supports service worker generation

Opt-in, requires `@angular/service-worker`

When activated, service worker generation is automatically added to the `ng build` command (for production builds)

Static Caching

```
{
  "index": "/index.html",
  "assetGroups": [
    // …
  ],
  "dataGroups": [
    // …
  ]
}
```

Name of index document

Caches source files of a single app version (i.e., static caching)

Survive app versions (i.e., for dynamic caching)*

* we recommend using IndexedDB instead

Static Caching

```
{
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": []
      }
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [],
        "urls": []
      }
    }
  ]
}
```

Default is prefetch

Prefetch: Cache directly

Specify files or urls

Lazy: Cache when requested

Dynamic Caching

Caching Strategies

```
"dataGroups": [
  {
    "name": "my-api",
    "urls": ["/api"],
    "cacheConfig": {
      "strategy": "freshness",
      "maxSize": 30,
      "maxAge": "30m",
      "timeout": "2s"
    }
  }
]
```

Maximum amount of cached requests

Only for freshness

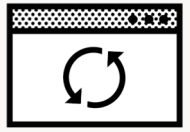
freshness: network-first
performance: cache-first

Maximum cache duration

PWA Features

Fresh

Fresh



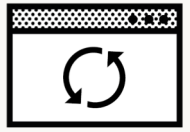
Keep your app up-to-date

The Service Worker caches a specific version of your application.

If the user is online, you want to deliver the latest version of your app.

When new versions become available, you might want to notify the user to update the application.

Fresh

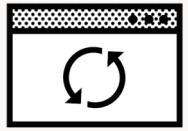


A regular service worker is updated when a new service worker script is deployed and fetched by the browser (at least once a day)

As the Angular Service Worker implementation barely changes, its update behavior is different: it uses a cache manifest instead

Fresh

@angular/service-worker Update Process



On every page reload, the `@angular/service-worker` reads the `ngsw.json` cache manifest generated by the CLI as a part of the build process.

This file contains information about the bundled files and their contents.

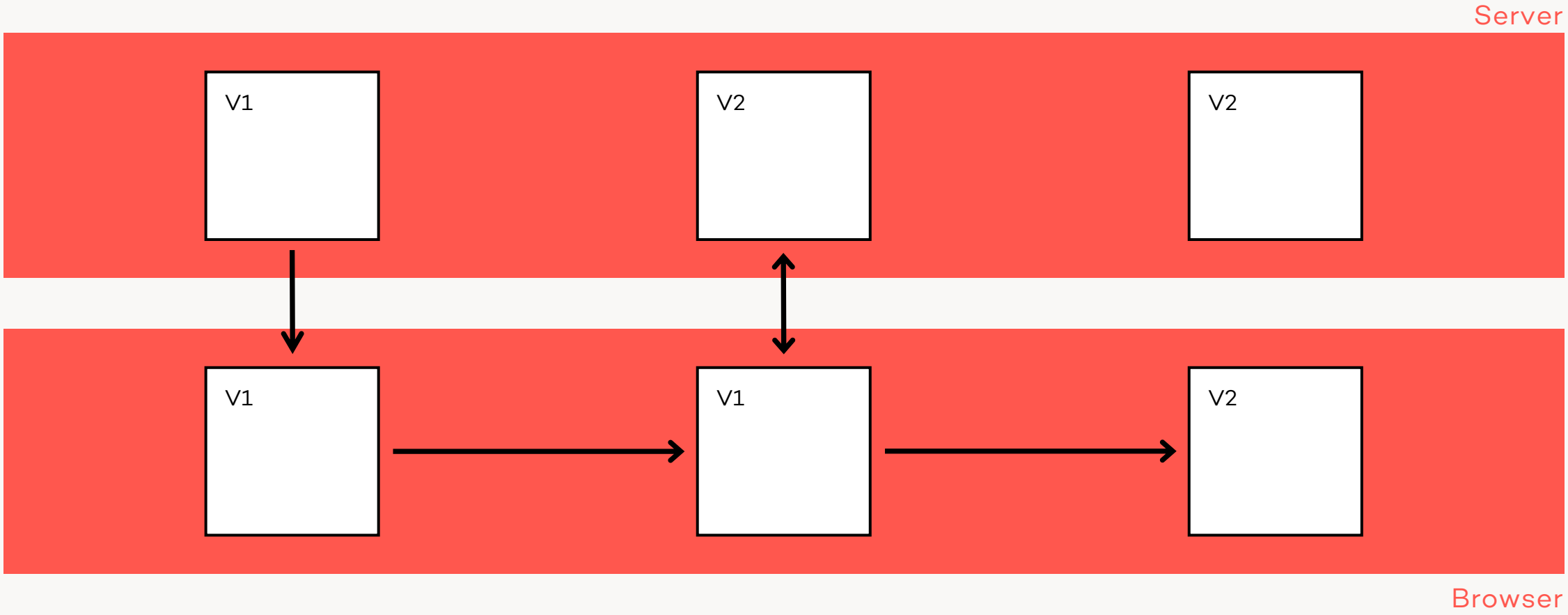
If this file differs from the previous version, the Service Worker will refresh your app.

Hence, it's sufficient to just reload the application after an update.


```
{
  "configVersion": 1,
  "index": "./index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "updateMode": "prefetch",
      "urls": [
        "./favicon.ico",
        "./index.html",
        "./0.7e11a8d89df9202c1c46.chunk.js",
        "./inline.2d50cafb9bf29e23968d.bundle.js",
        "./main.9c09878c6d7f0e439978.bundle.js",
        "./polyfills.4daeec822fa401be7786.bundle.js",
        "./styles.b49930bb2462609f4c36.bundle.css"
      ],
      "patterns": []
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "urls": [...],
      "patterns": []
    }
  ],
  "dataGroups": [],
  "hashTable": {
    "./0.7e11a8d89df9202c1c46.chunk.js": "b5297508c5a3e8e755ee522e4a540045f716cf81",
    "./assets/apple-touch-icon.png": "ab482b2ade7b58a2492cf846c3d518a0e1c99416",
    "./inline.2d50cafb9bf29e23968d.bundle.js": "cca79ae1b612a9665d88e18505de8c28a5e380fb",
    "./assets/icon-1024x1024.png": "8b61d8d837ec61a57039662030769c96fe7e2e9e",
    "./main.9c09878c6d7f0e439978.bundle.js": "7aa51186828042c21521b9221dfaf4de96b22c52",
    "./assets/icon-144x144.png": "09899f02788acf51f73dd2fbfc440e43f9d0a2d8",
    "./polyfills.4daeec822fa401be7786.bundle.js": "c043527e3dc1b73d0c6a21a91b41b735aac7fac5",
    "./assets/icon-512x512.png": "354e5f5fafd703db93923c7109992f8e661515d6",
    "./styles.b49930bb2462609f4c36.bundle.css": "43e317022acf4c217458855fdb774302367966ad",
    "./assets/logo.svg": "b5761f38a5d997eb71896aced8a85e63d2038b5d",
    "./favicon.ico": "734c4528daed4edeacbf46f34099a2c0bc0c1172",
    "./assets/mstile-150x150.png": "752a77079f6f62db223a9302a341d956ee5d9d97",
```

Fresh

@angular/service-worker Update Process



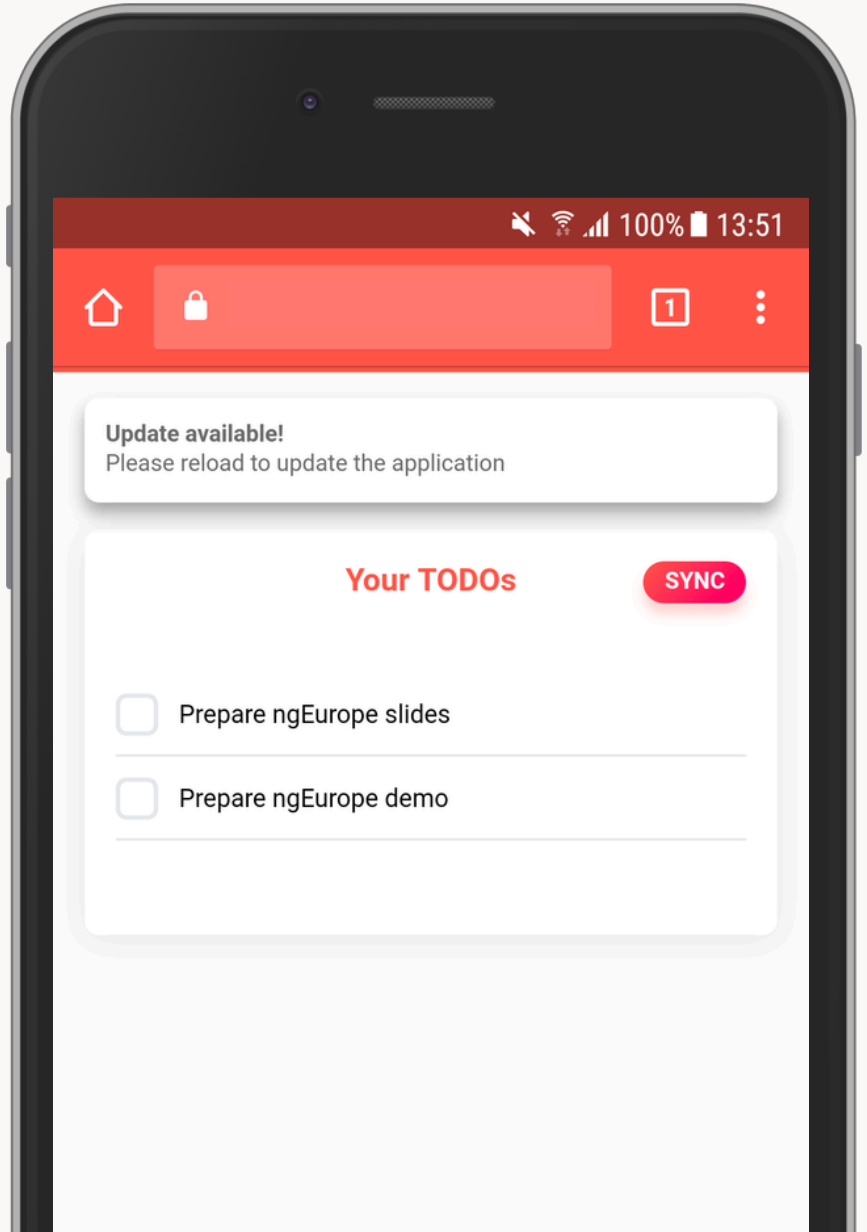
Fresh

SwUpdate

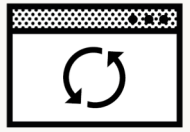
`@angular/service-worker` offers an `SwUpdate` service

This service provides an `available` observable that fires when there's an update.

As a result, the user can be prompted to reload the website.



Static Caching & SwUpdate



App-like, Connectivity Independent, Fresh

Cache external content

via `ngsw-config.json`

```
"assetGroups": [{ /*…*/ "resources": { "urls": [] } /*…*/ }]
```

Check for application updates

```
swUpdate.available.subscribe(() => …);
```

LAB #3

PWA Features

Connectivity Independent

Part 2

Offline Capability

Orthogonal to Service Workers

The web platform has different techniques to store arbitrary data (e.g. application state) offline:

- Web Storage API (Local Storage/Session Storage—synchronous!)
- Cookies (inconvenient)
- IndexedDB

Offline Capability

IndexedDB

Indexed Database (IndexedDB) is a database in the browser capable of storing structured data in tables with keys and value

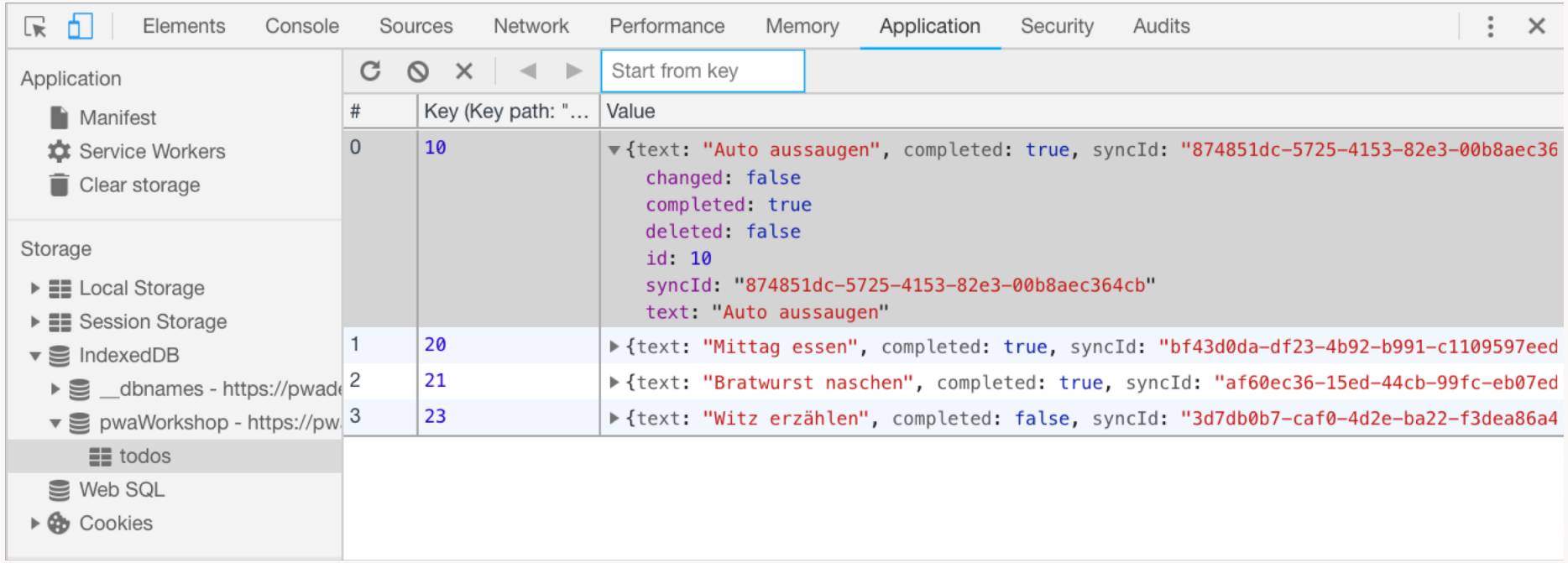
Data is stored permanently (survives browser restarts)

Service Worker and website share access to the same IndexedDB database (e.g. for synchronization purposes)

IndexedDB can help when storing client data, whereas Service Worker helps caching server data

Offline Capability

IndexedDB



Offline Capability

IndexedDB

11

4

7.1

10

Offline Capability

IndexedDB

```
let db;
const request = indexedDB.
  open('TestDB');

request.onerror = err => {
  console.error('boo!');
};

request.onsuccess = evt => {
  db = request.result;
};
```

IndexedDB API is inconvenient

```
const db =
  new Dexie('TestDB');
```

Offline Capability

Dexie.js

Dexie is a minimalistic wrapper for IndexedDB

Operations are based on promises instead of callbacks

Near native performance (also for bulk inserts)

Open-source

Offline Capability

Dexie.js Table API


```
const table = db.table('todos');

table.toArray()                    // get all items as an array
table.add()                        // insert an object
table.put()                        // update or insert an object
table.filter(i => i.id !== 3)      // apply JS filter on value
table.where({ name: 'Peter' })     // query items by key
```

Offline Capability

Dexie.js Table API


```
public getAll(): Promise<Array<TodoItem>> {
  return table.toArray();
}
```

Remember: APIs return Promises

Many parts of Angular use Observables instead of Promises

Offline Capability

await/async Keywords

```
function x() {
  table.toArray()
    .then(t => t.forEach(i => console.log(i)))
}
```

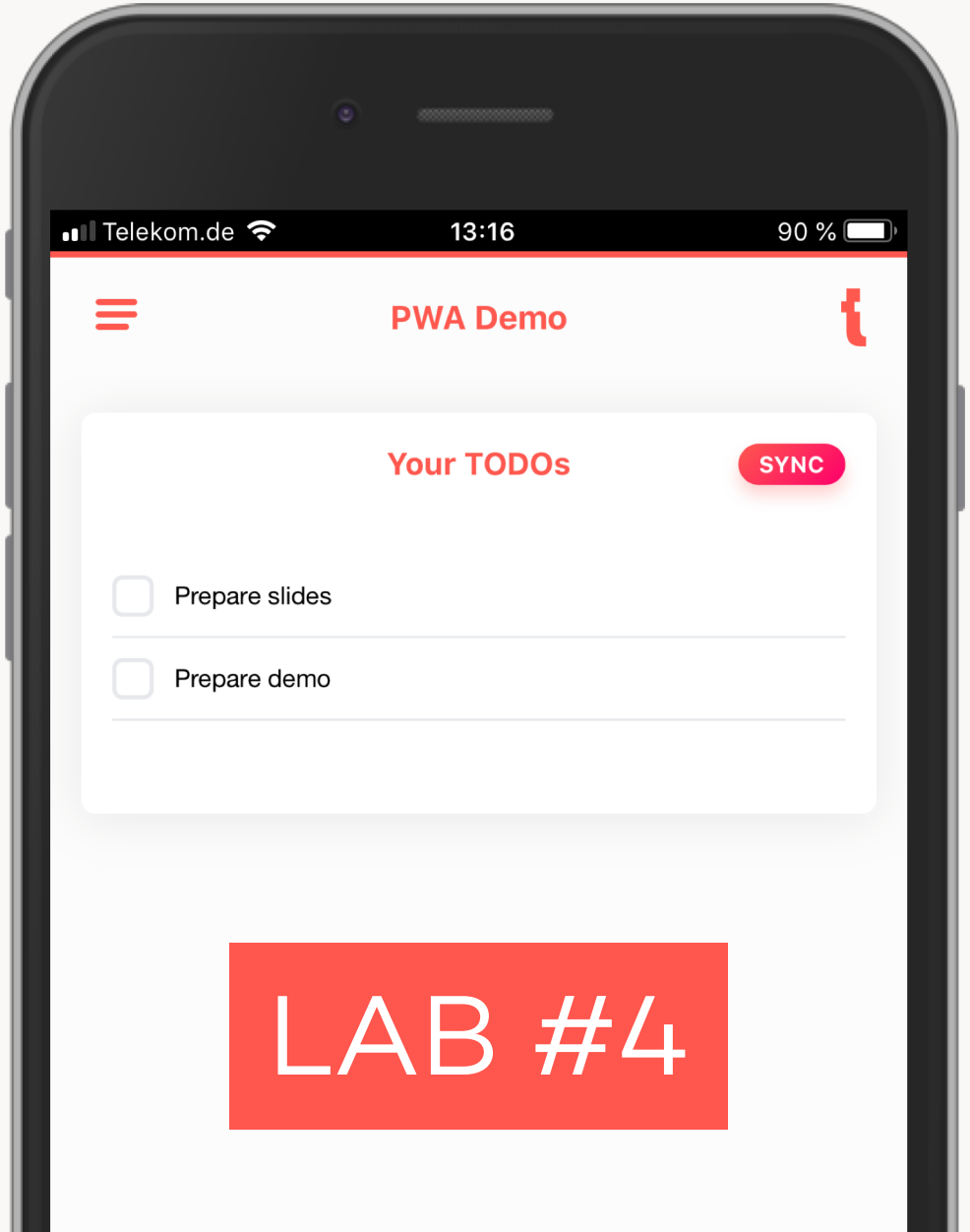
```
async function x() {
  const t = await table.toArray();
  t.forEach(i => console.log(i));
}
```

Offline Capability

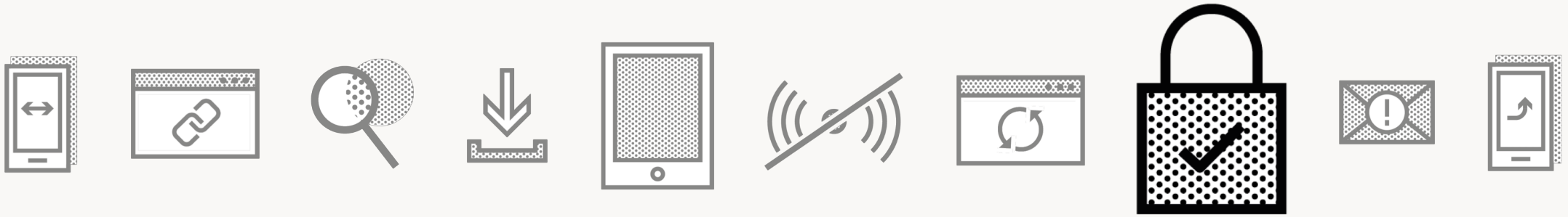
IndexedDB

Dexie.js Table API

Implement the `getAll`, `add`, `update`, `delete` methods of `TodoService`



PWA Features



Safe

Safe

PWAs Require HTTPS!

Service workers are very powerful

- Can answer representatively for the server
- Runs outside of the tab's/window's lifecycle

Installation of service workers is only allowed via a secure connection

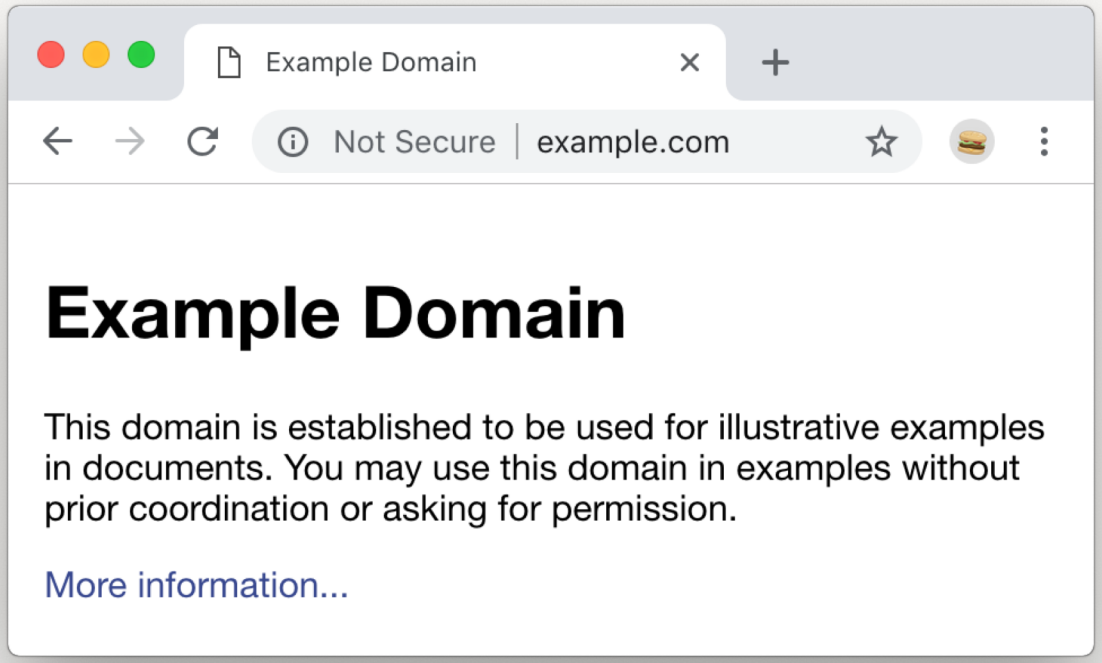
Exception: localhost (for development purposes)

PWAs can only be delivered via HTTPS

Problem: TLS certificates are required (costs/renewal)

Safe

HTTP sites are “Not Secure” in Chrome 68+



Safe

PWAs Require HTTPS!

	Recurring Costs	Free
Manual Renewal Required	Comodo GeoTrust Verisign DigiCert ...	?
Automatic Renewal	?	Let's Encrypt Third-Party (Azure, GitHub, CloudFlare)

Let's Encrypt

Free with automatic renewal

Initiative of the Linux Foundation
Automatic domain validation (Class 1)
Website/Webservice has to be public available
Period of validity: Comparatively short (90 days)

Usage of TLS/SSL as easy as possible

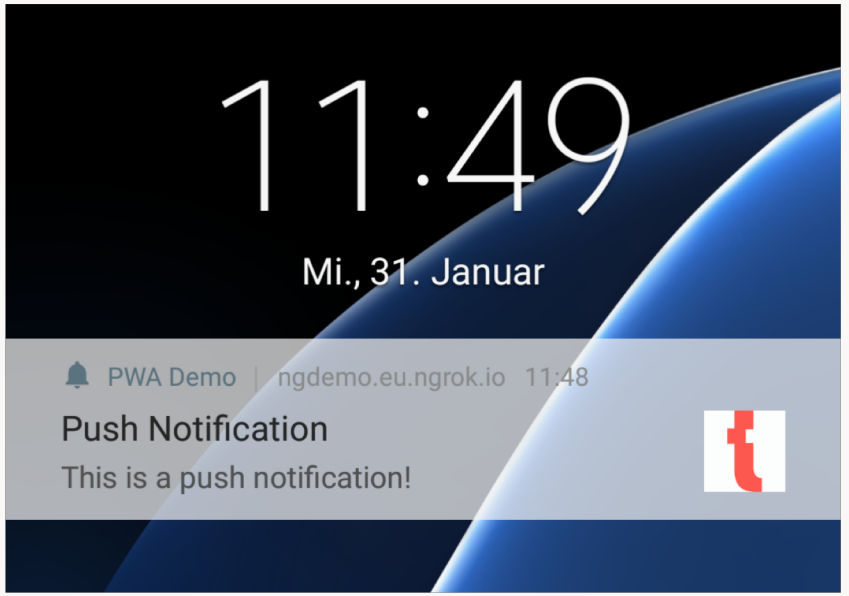
Plugins are available for IIS, Apache, Microsoft Azure, Plesk, ... free!

PWA Features

Re-engageable

Re-Engageable

Get the user back with notifications



Idea: Push the users to use the app again

Known from social networks or games, etc.

Combination of two technologies:

- Web Notifications
- Push API

Re-Engageable

Web Notifications

5

22

6
(macOS)

14

Re-Engageable

Push API

44

44

—

17

Re-Engageable

Push services

Every browser vendor has its own push service

Challenge: Every service is used in a different way

Ends in several implementations on the server

Solution: One protocol that can be used by every push service

Web Push Protocol

Re-Engageable

Web Push Protocol

Every push service talks Web Push Protocol

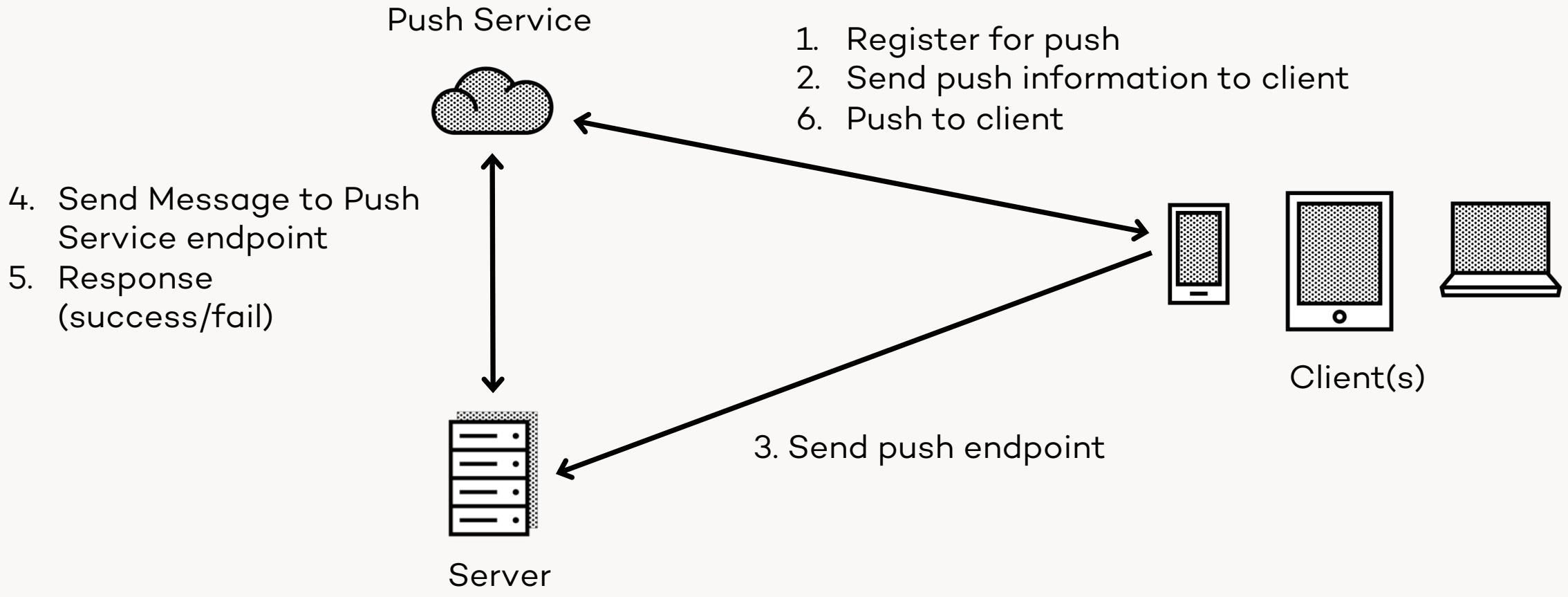
Only one server implementation needed

To make the push request secure, private and public keys are needed

Ready-to-use-packages are available for different servers (e.g. Node, ASP.NET, etc.)

Re-Engageable

Sending Push Notifications



Re-Engageable

SwPush

`@angular/service-worker` offers an `SwPush` service

The service provides a `requestSubscription` promise that returns a new push subscription

This subscription includes the push service endpoint

That endpoint can be used from the server to send push notifications to the client

Push Notifications

Lab

LAB #5

Create key pair

https://web-push-codelab.glitch.me/ can be used to generate key pairs

Update server

Register for Push

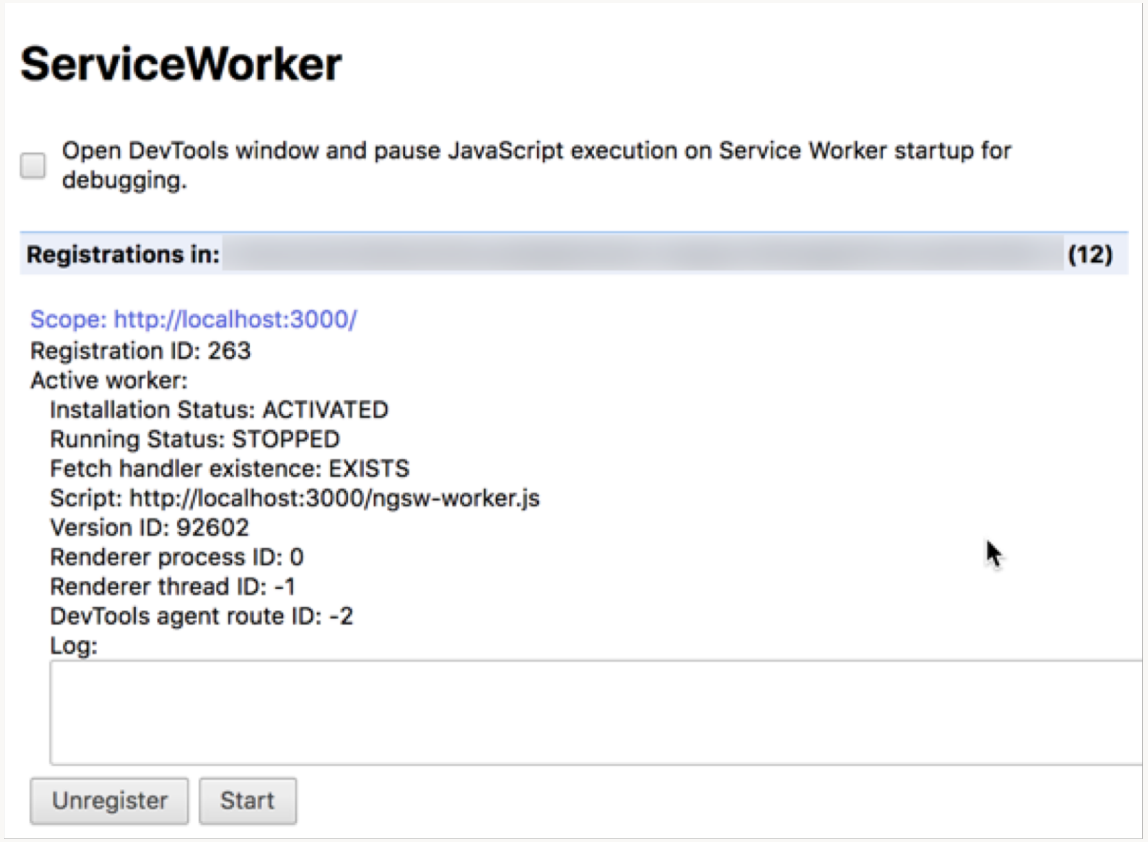
```
const subscription = await swPush.requestSubscription({
serverPublicKey: "public key"});
```

Service Worker

Debugging

More information on installed service workers can be found on

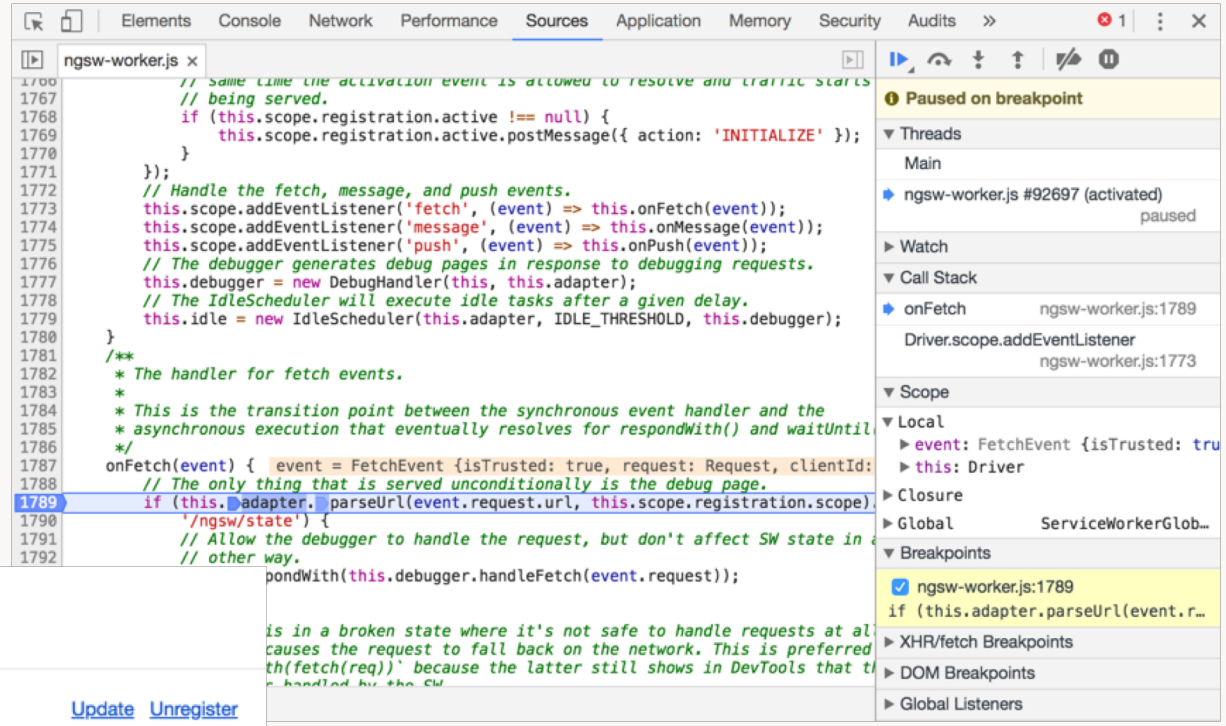
- `chrome://serviceworker-internals` (Google Chrome)
- `about:serviceworkers` (Mozilla Firefox)

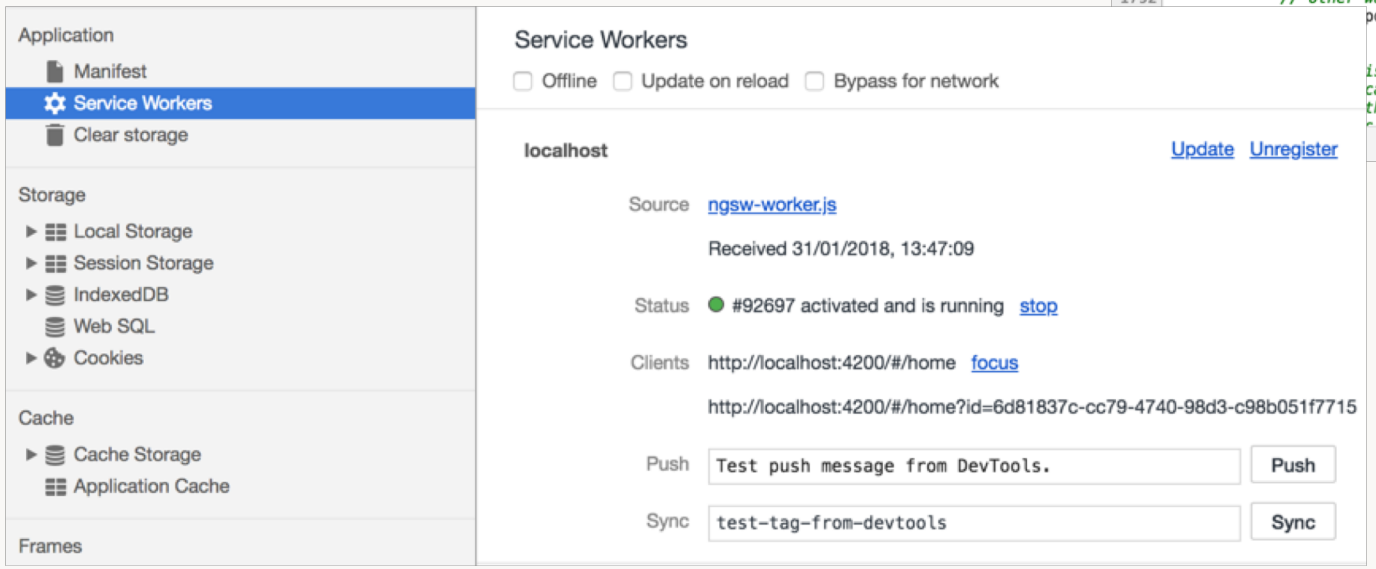


Service Worker

Debugging

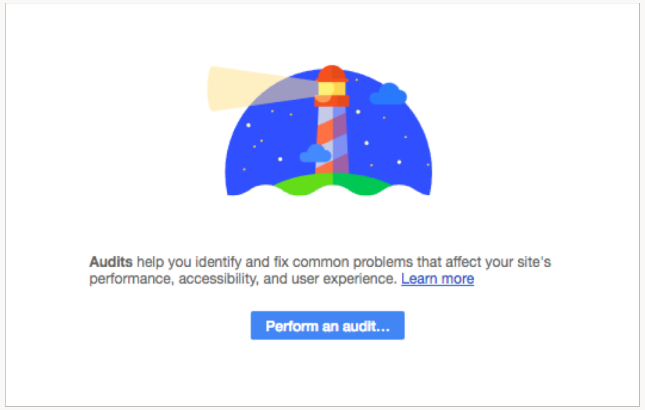
Debug Service Workers using Chrome Developer Tools

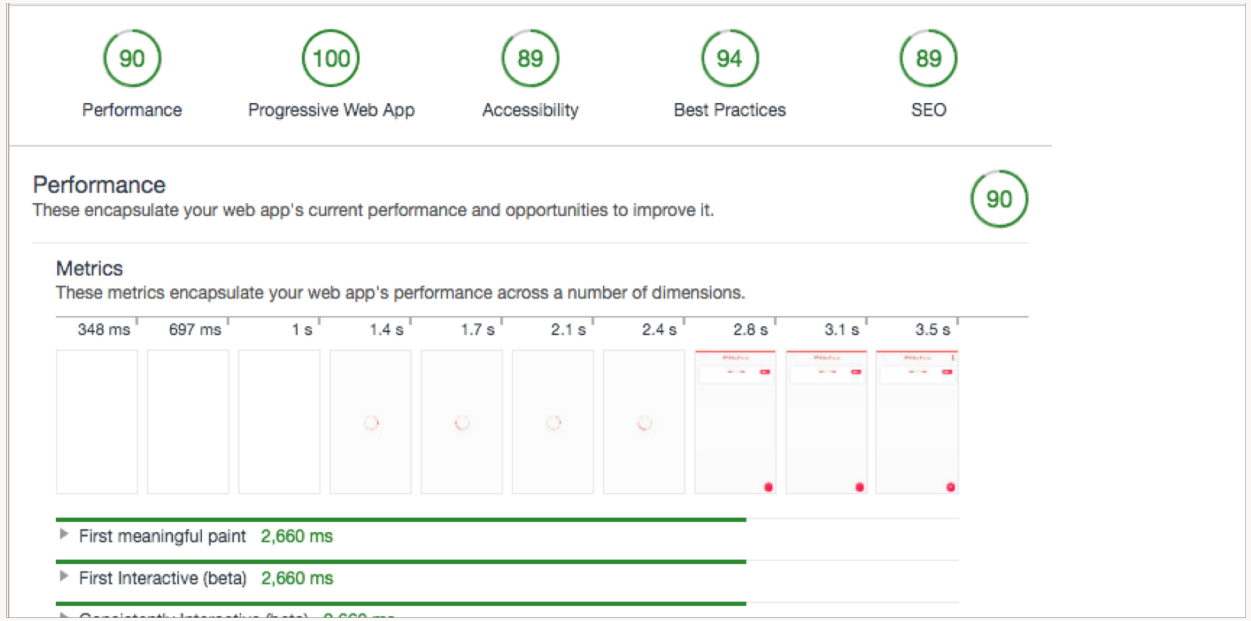




Tools

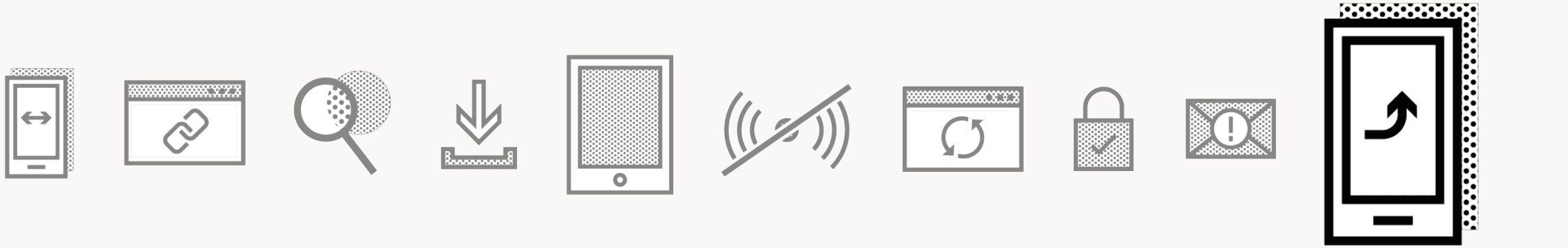
Lighthouse/Audits





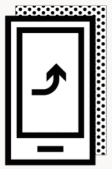
LAB #6

PWA Features



Progressive

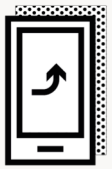
Progressive \prə-ˈgre-siv\



- moving forward
- happening or developing gradually over a period of time
- using or interested in new or modern ideas especially in politics and education

Source: Merriam-Webster's Learner's Dictionary

Progressive



Advance with Progressive Enhancement

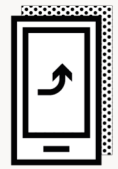
Idea: Use available interfaces and functions of a system

Users with modern, feature-rich browsers are getting better experience

Apps are available on older browsers but with limited functionality

Concept: Browser feature support grows over time—thereby more users can enjoy an increasing number of app features

Progressive



Advance with Progressive Enhancement

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register(…)
        .then(() => {
            /* … */
        });
}
```

In JavaScript: check whether or not an API/feature is available

If yes—use it!

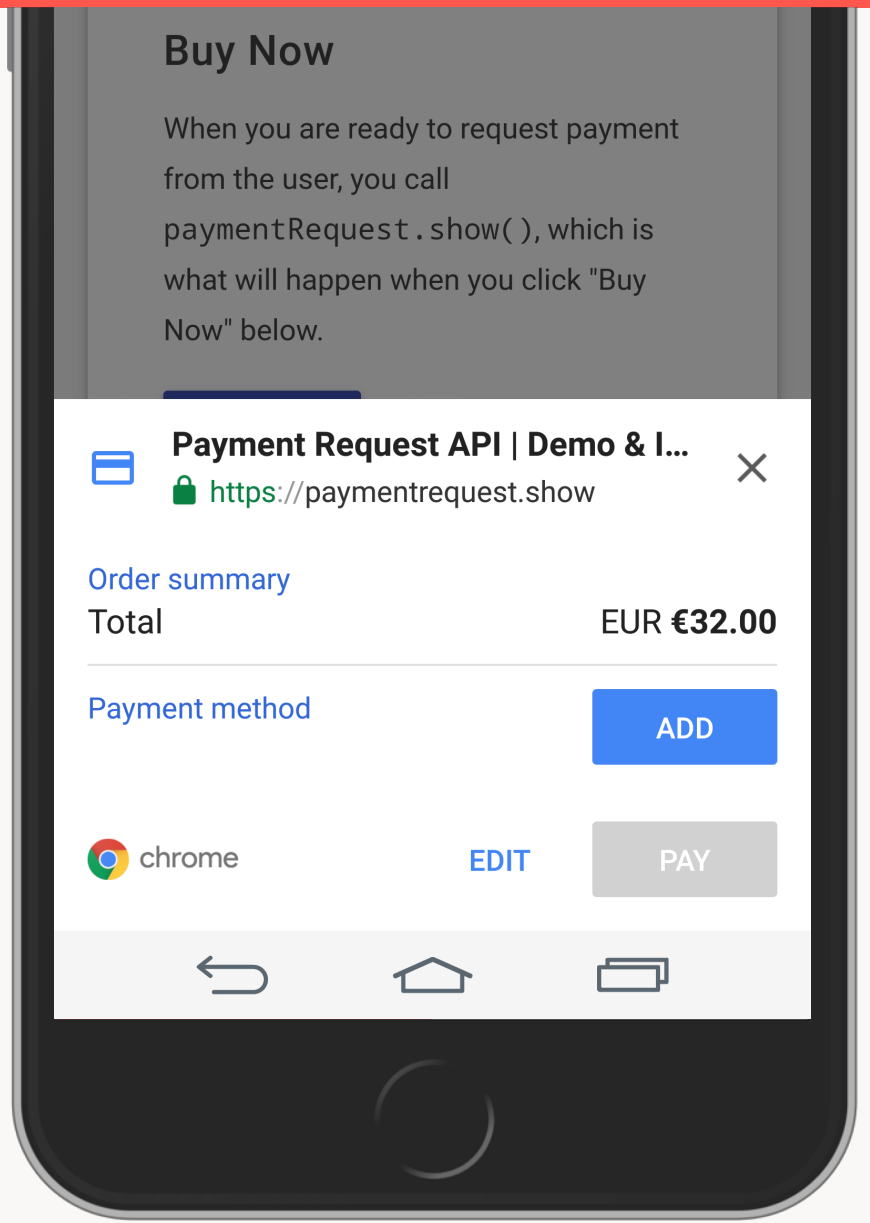
Progressive

Monetization w/ Payment Request API

Upcoming payment technology for the web

Supported by Microsoft Edge 15+, Google Chrome 61+, Safari 11.1+ (iOS 11.3+)

https://blogs.windows.com/msedgedev/2016/12/15/payment-request-api-edge/



Progressive

Payment Request API

61

64

11.1

15

Progressive

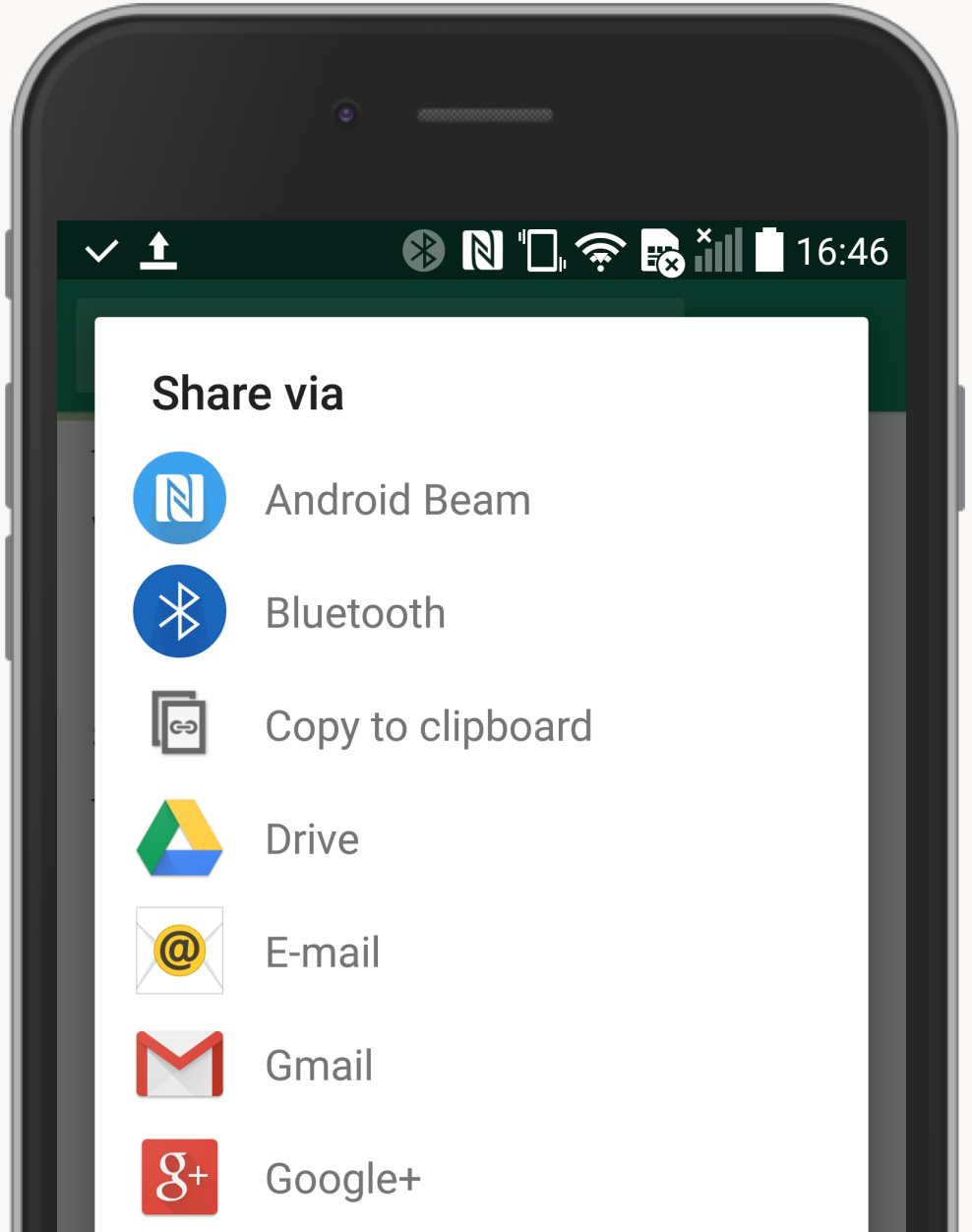
Web Share API

Relatively new API of the web platform

Allows sharing any URLs or text

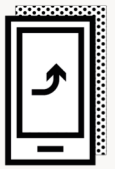
Source website must be served over HTTPS

API can only be invoked as a result of a user action (i.e. a click)



Progressive

Web Share API



69
(Android)

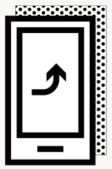
?

In
Preview

Under
Consideration

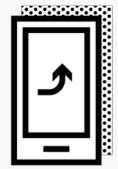
Progressive

Web Share API



```
navigator.share({
  title: 'title',
  text: 'text',
  url: 'https://foobar.com'
})
  .then(() => /* success */)
  .catch(err => /* error */);
```

Progressive



Web Share API

In our demo application, there's a `FeatureService` for detecting browser features (Progressive Enhancement)

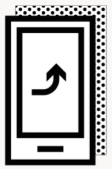
Implement Web Share API in `share.service.ts`

Use a fallback method in case Web Share API is unavailable

LAB #7

Progressive

PRPL Pattern



Push critical resources for the initial URL route

Render initial route

Pre-cache remaining routes

Lazy-load and create remaining routes on demand

https://developers.google.com/web/fundamentals/performance/prpl-pattern/

Progressive

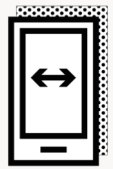
Server-side rendering w/ Anguar Universal

LAB #8

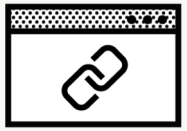
In order to reduce the **time to interactive (TTI)**, you can use Angular Universal to pre-render the app shell. This reduces the TTI especially on mobile devices.

```
ng g appShell
  --client-project=sample-app
  --universal-project=server-app
```

“Uber Pattern”



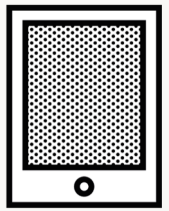
Responsive



Linkable

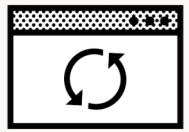
Discoverable

Installable



App-like

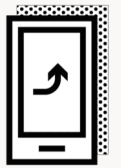
Connectivity Independent



Fresh

Safe

Re-engageable



Progressive

Workbox—an Alternative

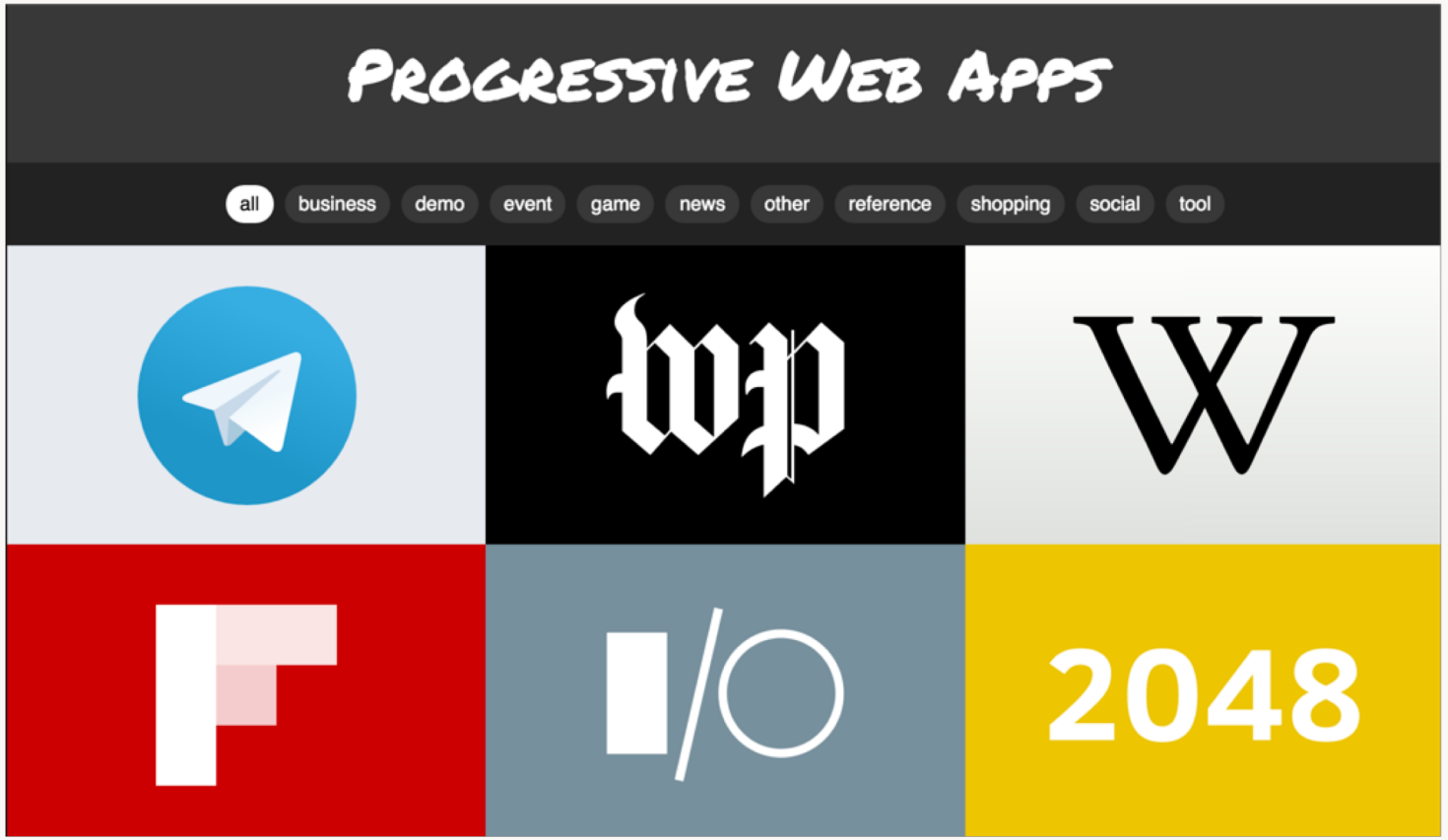
NgServiceWorker

- One-size-fits-all Service Worker
- Angular CLI supports Service Worker generation
- Pre-defined functionality, limited customizability

Custom Implementation

- Individually created Service Worker
- Post-process build output of an Angular app
- Maximum flexibility, maximum management overhead

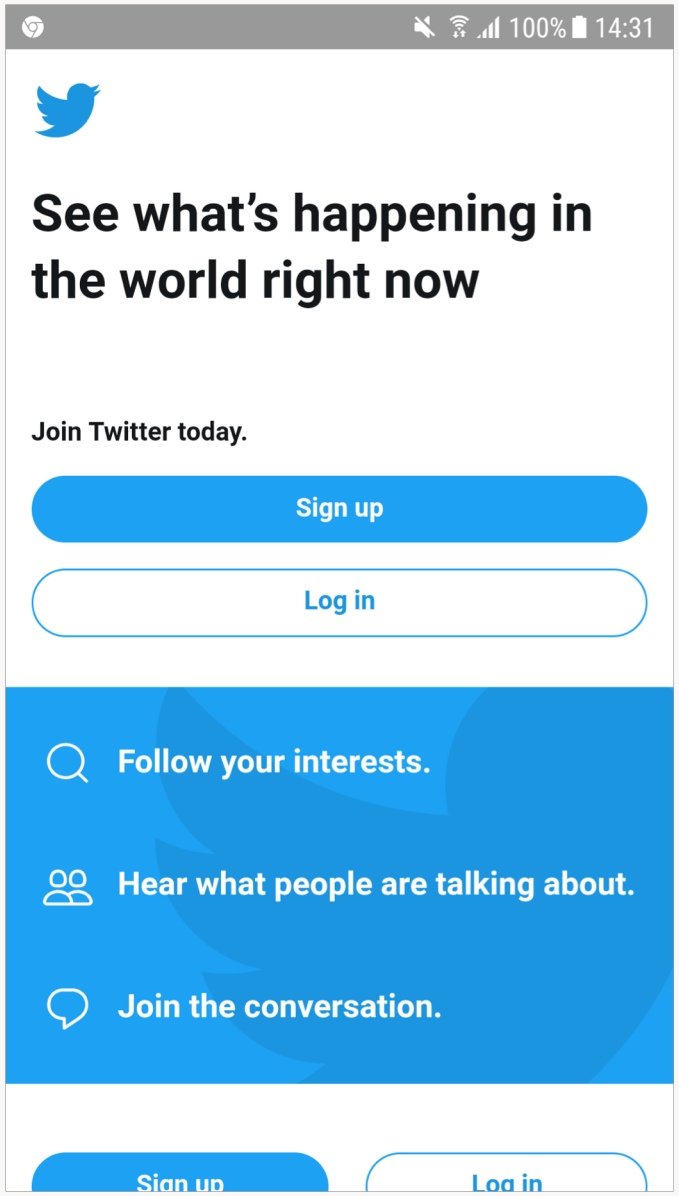
Reference Apps



https://pwa.rocks

Reference Apps

mobile.twitter.com



Thank you

for your kind attention!

think tecture

Steffen Jahr
@steffenjahr
steffen.jahr@thinktecture.com

Christian Liebel
@christianliebel
christian.liebel@thinktecture.com