

PWA

Progressive Web Apps

WJAX 2019



Karsten Sitterberg

- Entwickler, Trainer, Berater
 - @kakulty
 - <https://sitterberg.com>
- Java User Group Münster
- Frontend Freunde Münster



Vorteile: Native App

- Offlinefähig
- Installierbar
- Zugriff auf reichhaltige APIs

Web

- Web ist offener Standard
- Deployment einfach
- Nicht installierbar, nicht offlinefähig
- Kein Zugriff auf System-APIs
- **Abhilfe: Progressive Web Apps**

Installierbar

Offline fähig

HTML5 APIs

FIRE

Fast

App reagiert auf Nutzer-Eingaben **direkt** (Button-Toggle, Input, ...)

Integrated

Auf Device **installierbar** (wie native App)

Reliable

App lädt und öffnet **direkt**, auch wenn **offline**

Engaging

Interaktiv durch Ausnutzung des Devices (Push, Sensors, Kamera, ...)

Aktueller Plattform Support (Android 9, iOS 13)

	Android	iOS
Service Worker	✓	✓
SW-Caching	✓	✓
Push	✓	✗
Sync	✓	✗
Manifest	✓	✓

Progressive

Einsatzbereich

Google-Suche umgestellt: PWA

Statt Native-Apps: Web Apps (Pizza bestellen, ...)

Ausgangsbasis für Native-App



Einsatzbereich

PWA ist für **jeden!**

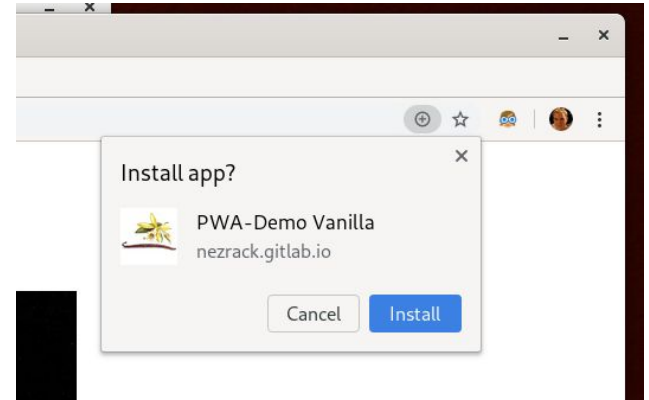


Aber: Unterschiedliche “Nutzungstiefe” der Features

Google-Suche umgestellt: PWA

Statt Native-Apps: Web Apps (Pizza bestellen, ...)

Ausgangsbasis für Native-App



Verwendete Software

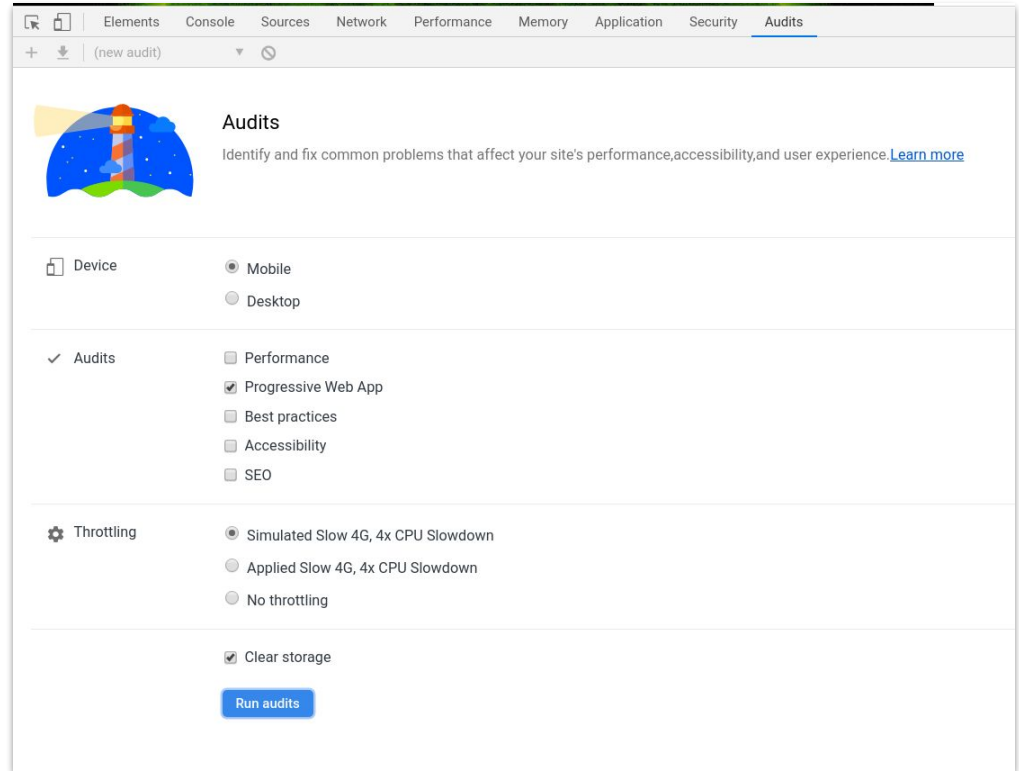


Angular



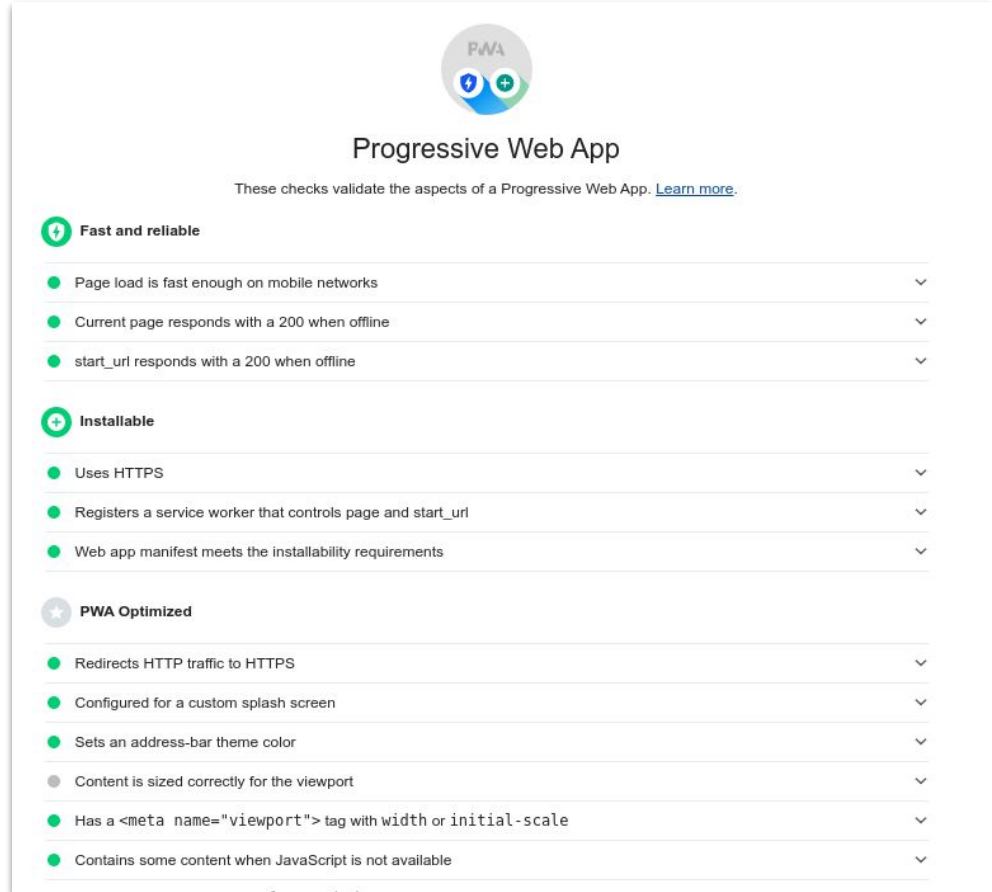
Chrome Audits

- Ehemals “Lighthouse”



Chrome Audits

- Ehemals “Lighthouse”



The screenshot displays the 'Progressive Web App' audit section in Chrome DevTools. At the top, there is a 'PWA' icon with a lightning bolt and a plus sign. Below it, the title 'Progressive Web App' is centered, followed by a subtitle: 'These checks validate the aspects of a Progressive Web App. [Learn more.](#)'

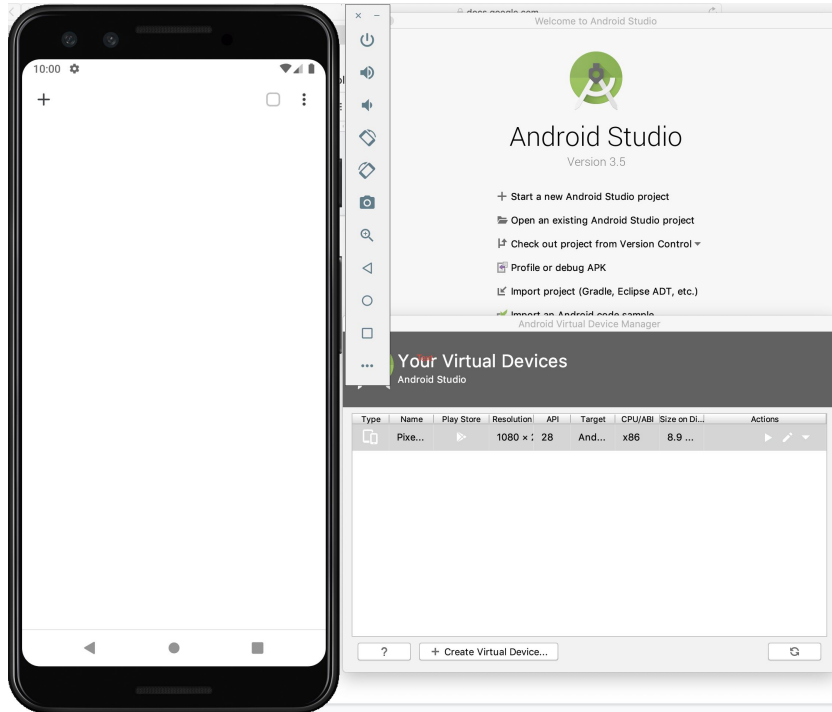
The audit is organized into three main categories, each with a green circular icon:

- Fast and reliable** (lightning bolt icon):
 - Page load is fast enough on mobile networks
 - Current page responds with a 200 when offline
 - start_url responds with a 200 when offline
- Installable** (plus icon):
 - Uses HTTPS
 - Registers a service worker that controls page and start_url
 - Web app manifest meets the installability requirements
- PWA Optimized** (star icon):
 - Redirects HTTP traffic to HTTPS
 - Configured for a custom splash screen
 - Sets an address-bar theme color
 - Content is sized correctly for the viewport
 - Has a `<meta name="viewport">` tag with width or initial-scale
 - Contains some content when JavaScript is not available

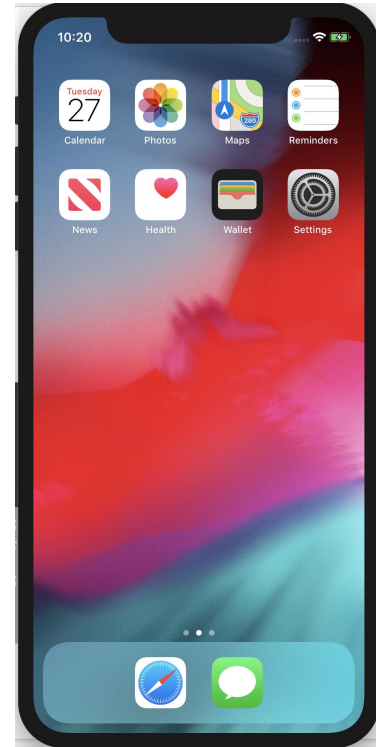
Each individual check is preceded by a green dot, indicating a passing status. Each check also has a downward-pointing chevron icon on its right side, suggesting expandable details.

Emulators

Android Studio AVD



XCode iPhone-Simulator





GitLab

GitLab Pages

- Automatischen Deployment via GitLab-CI
- Voraussetzungen
 - `.gitlab-ci.yml`
 - CI-Task, der Dateien in `"/public"`-Ordner kopiert

GitLab Pages

The screenshot shows the GitLab navigation sidebar. At the top, the GitLab logo and navigation links for 'Projects', 'Groups', 'Activity', and 'Milestones' are visible. Below this, the sidebar is organized into sections: 'Project' (pwa-demo), 'Details' (Activity, Releases, Cycle Analytics), 'Repository' (Issues, Merge Requests, CI / CD), 'Operations' (Operations, Packages, Wiki, Snippets), and 'Settings'. The 'Settings' menu is expanded, showing options like General, Members, Integrations, Repository, CI / CD, Operations, Pages (highlighted), and Audit Events.

Karsten Sitterberg > pwa-demo > Pages

Pages

New Domain

With GitLab Pages you can host your static websites on GitLab. Combined with the power of GitLab CI and the help of GitLab Runner you can deploy static pages for your individual projects, your user or your group.

Force HTTPS (requires valid certificates)

Save

Access pages

Congratulations! Your pages are served under:

<https://nezrack.gitlab.io/pwa-demo>

Remove pages

Removing pages will prevent them from being exposed to the outside world.

Remove pages

GitLab Pages

...

```
pages:  
  stage: deploy  
  script:  
    - mkdir public  
    - cp -r $DIST_DIR/* public  
  artifacts:  
    paths:  
    - public  
  only:  
    - master
```

HTML 5 APIs

Auch für "normale" Webseiten

Achtung

HTTPS nutzen

Mit localhost/127.0.0.1 reicht (meist) HTTP

Aufpassen mit Incognito/Private Browsing

Funktionen teils nicht verfügbar!

Progressive Enhancement

```
if('serviceWorker' in navigator){ ... }
```

```
if('Notifications' in window) { ... }
```

```
if ('share' in navigator) { ... }
```

•

•

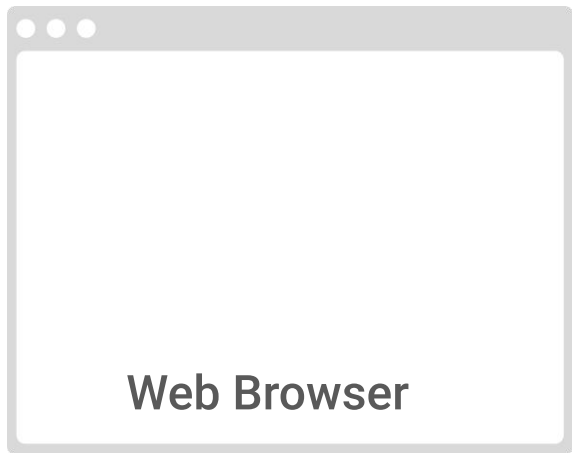
•

Offline First

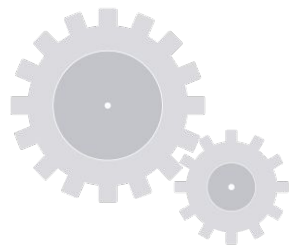
Anwendung für offline gebaut: **“Default”**

Online sein ist **“Feature”**

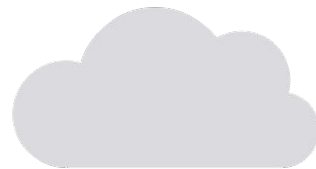
Service Worker



`/api/`
→

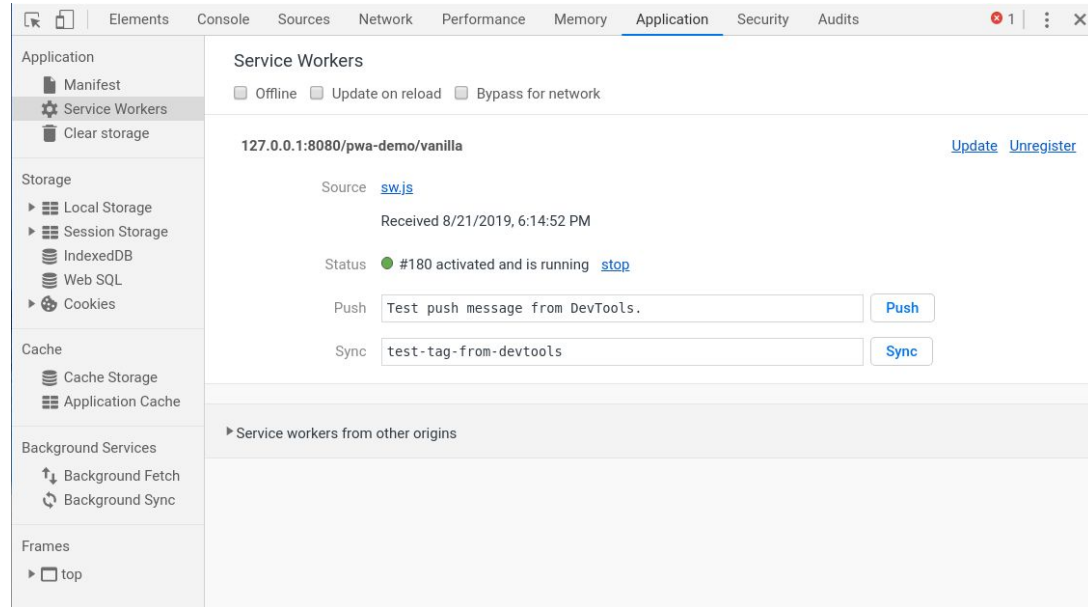


`/api/`
→



Service Worker

- Debugging in Chrome-Devtools
- SW stoppen/entfernen
- SW aktualisieren
- Cache leeren
- Push/Sync-Events auslösen



PWA-Manifest

PWA soll installierbar sein

Home-Icon

Splash-Screen

PWA soll installierbar sein

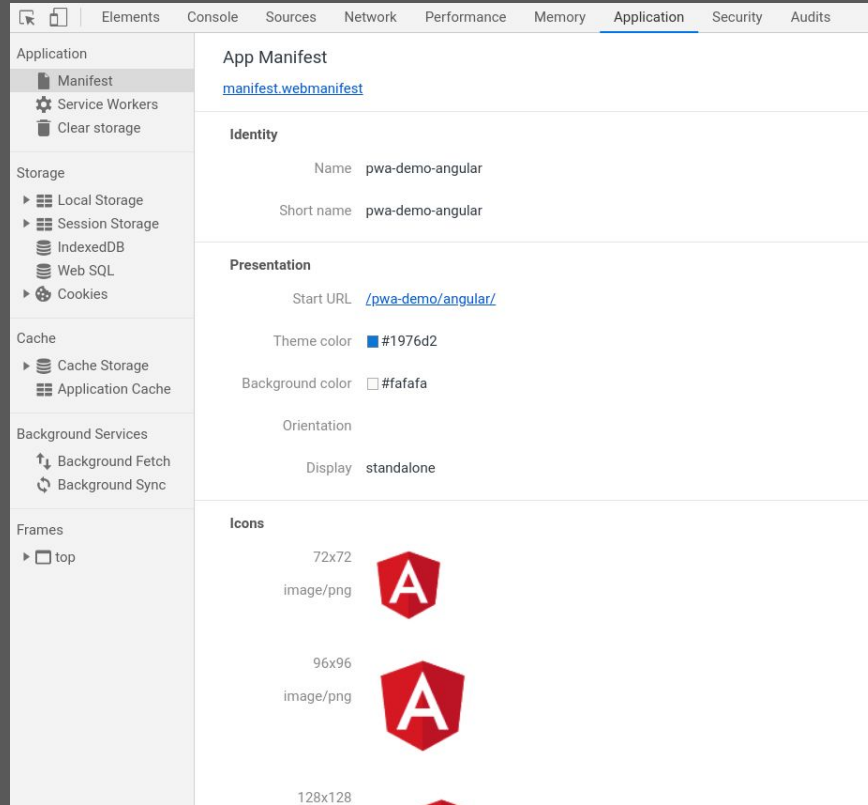


Web-Manifest

index.html




```
<head>  
  <link rel="manifest" href="manifest.webmanifest">  
  . . .  
</head>
```

In Chrome Dev-Tools



The screenshot shows the Chrome DevTools interface with the 'Application' tab selected. The left sidebar contains a tree view of application resources, including Manifest, Service Workers, Clear storage, Storage (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies), Cache (Cache Storage, Application Cache), Background Services (Background Fetch, Background Sync), and Frames (top).

The main content area displays the 'App Manifest' for the file `manifest.webmanifest`. The manifest details are as follows:

- Identity**
 - Name: `pwa-demo-angular`
 - Short name: `pwa-demo-angular`
- Presentation**
 - Start URL: `/pwa-demo/angular/`
 - Theme color: `#1976d2`
 - Background color: `#fafafa`
 - Orientation: (not specified)
 - Display: `standalone`
- Icons**
 - 72x72 image/png: 
 - 96x96 image/png: 
 - 128x128 image/png: 

Splash Screen

iOS: WebClip-Technologie (seit etwa **2008!**)

Mehrere **<meta>**- bzw. **<link>**-Angaben im HTML-Header erforderlich, z.B.:

```
<link rel="apple-touch-startup-image" href="/launch.png">  
<meta name="apple-mobile-web-app-title" content="AppTitle">  
<meta name="apple-mobile-web-app-capable" content="yes">
```



Pro iOS Device ein Icon

Angular PWA

Angular PWA



```
ng add @angular/pwa --project pwa-demo-ng
```

Installiert `@angular/service-worker`-Paket

Aktualisiert `index.html` mit **Manifest** und **Theme**

Registriert + Konfiguriert Serviceworker:

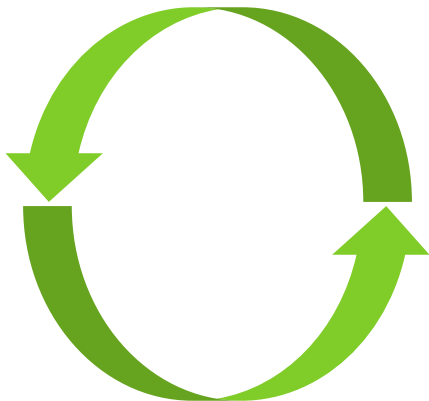
```
/src/app/app.module.ts
```

```
/ngsw-config.json
```


SwUpdate-Service

a. Event: **Erkennung** von App-Versionen

Auslieferung durch SW bei **Page-Refresh**

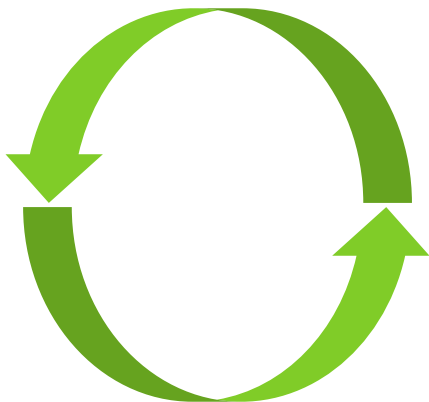


```
swUpdate.available.subscribe(event => {  
  console.log('Momentanversion:', event.current);  
  console.log('Verfügbar:', event.available);  
});
```

SwUpdate-Service

b. Event: **Aktivierung** einer Version

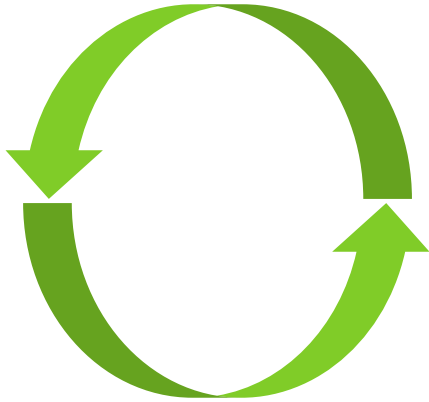
Service Worker liefert
neue Version **bei nächster Anfrage**



```
swUpdate.activated.subscribe(event => {  
  console.log('Alte Version', event.previous);  
  console.log('Neue Version', event.current);  
});
```

SwUpdate-Service

c. Server-Anfrage zur **Update-Erkennung** triggern



```
swUpdate.checkForUpdate()
```

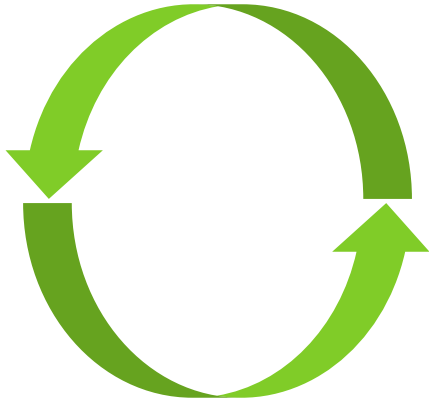
Oder simpel: **Page Reload**

SwUpdate-Service

d. Versions-Aktivierung triggern

Schaltet **Version im SW** "scharf"

Aktualisierung der App selbst: **Page-Reload**



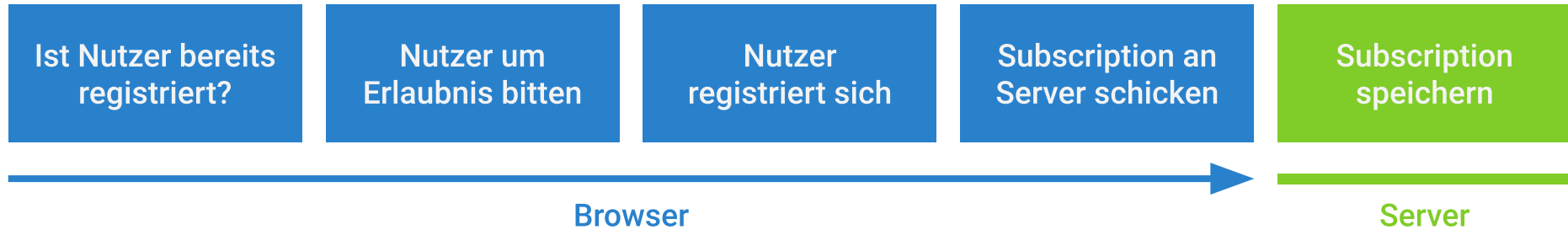
```
swUpdate.activateUpdate()  
  .then(() => document.location.reload());
```

Push-Notifications

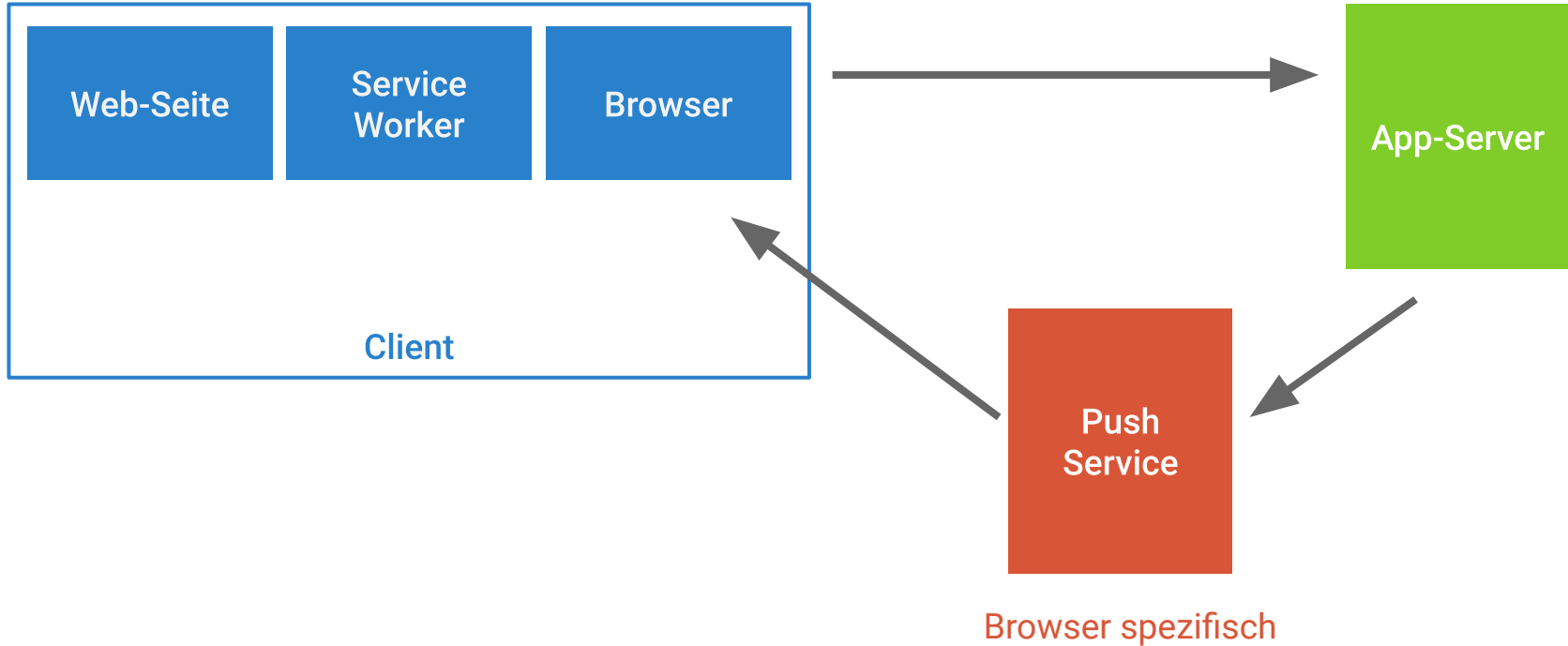
Push Notifications

1. **Notification:** Nachricht, angezeigt **außerhalb** des Browser
2. **Push Message:** Vom Server zum Client gesendete Nachricht
3. **Push Notification:** Als Resultat der Push Message angezeigte Notification

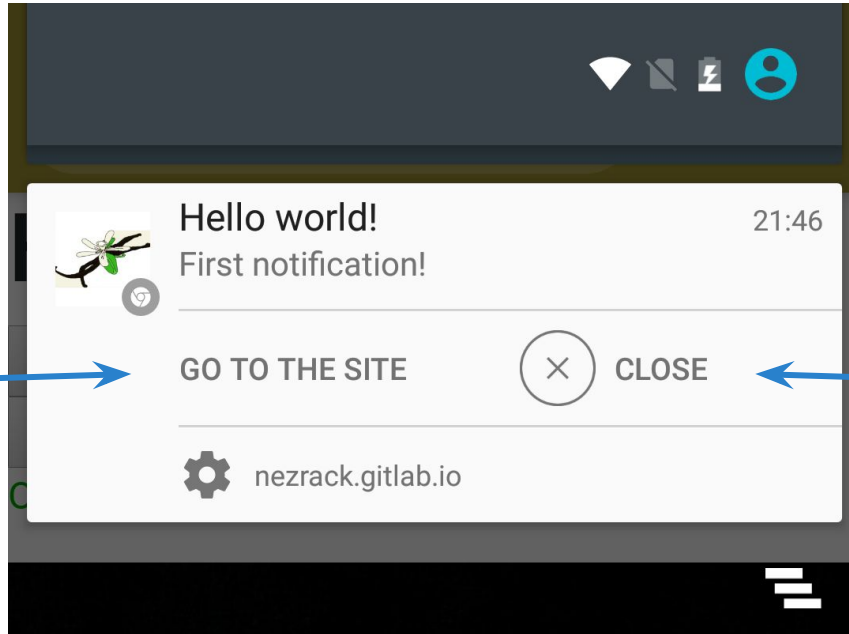
Subscribe auf Push-Messages



Architektur Push-Messages



Notification Actions



SW Push

SW-Push

Angular bringt eigenen Push-Service (**SwPush**) mit

Im Paket `@angular/service-worker`

```
import { SwPush } from '@angular/service-worker';  
/* ... */  
export class AppComponent {  
  constructor(readonly swPush: SwPush) {}  
/* ... */  
}
```

SW-Push

Reaktion auf ankommende Push: **Automatisch**

Stattdessen: Subscription auf Notification-Clicks

```
this.swPush.notificationClicks.subscribe(  
  ({action, notification}) => {  
    console.log(action, notification);  
  });
```

SW-Push

Notwendig dafür:

Spezielles Format der
Notification
(von Server zu Push-Service!)

```
{
  "notification": {
    "actions": NotificationAction[],
    "badge": USVString
    "body": DOMString,
    "data": any,
    "dir": "auto"|"ltr"|"rtl",
    "icon": USVString,
    "image": USVString,
    "lang": DOMString,
    "renotify": boolean,
    "requireInteraction": boolean,
    "silent": boolean,
    "tag": DOMString,
    "timestamp": DOMTimeStamp,
    "title": DOMString,
    "vibrate": number[]
  }
}
```

Angular App-Shell

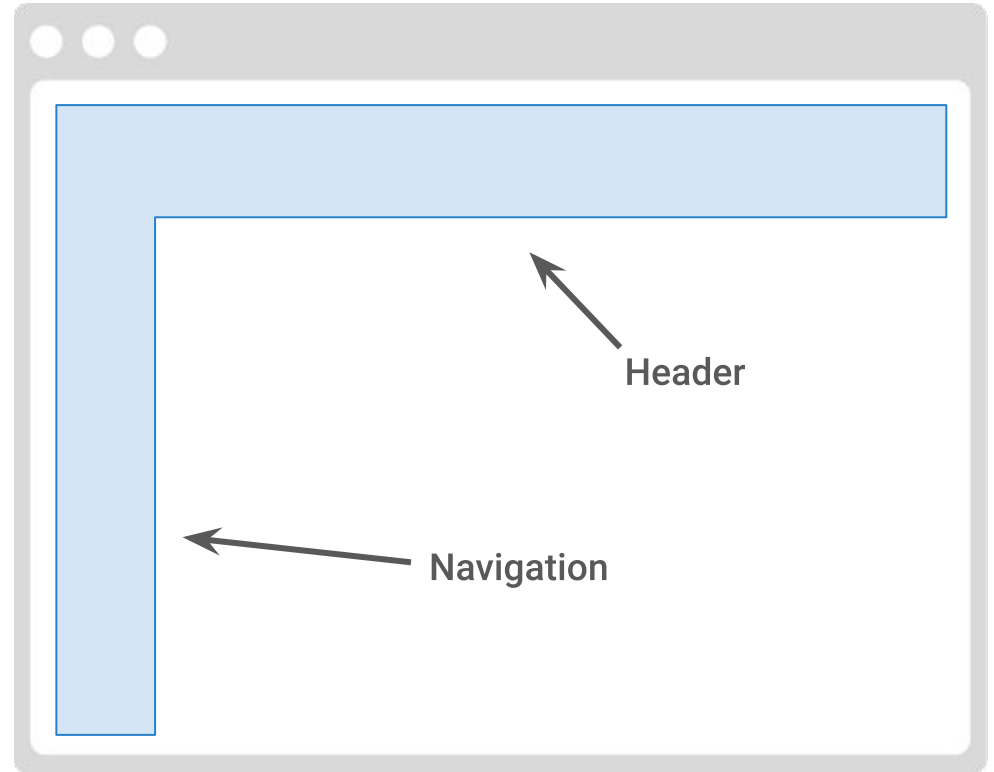
App-Shell

Minimales HTML/CSS/JS
(Statischer) Content

-> **Direkt da - dank Cache!**

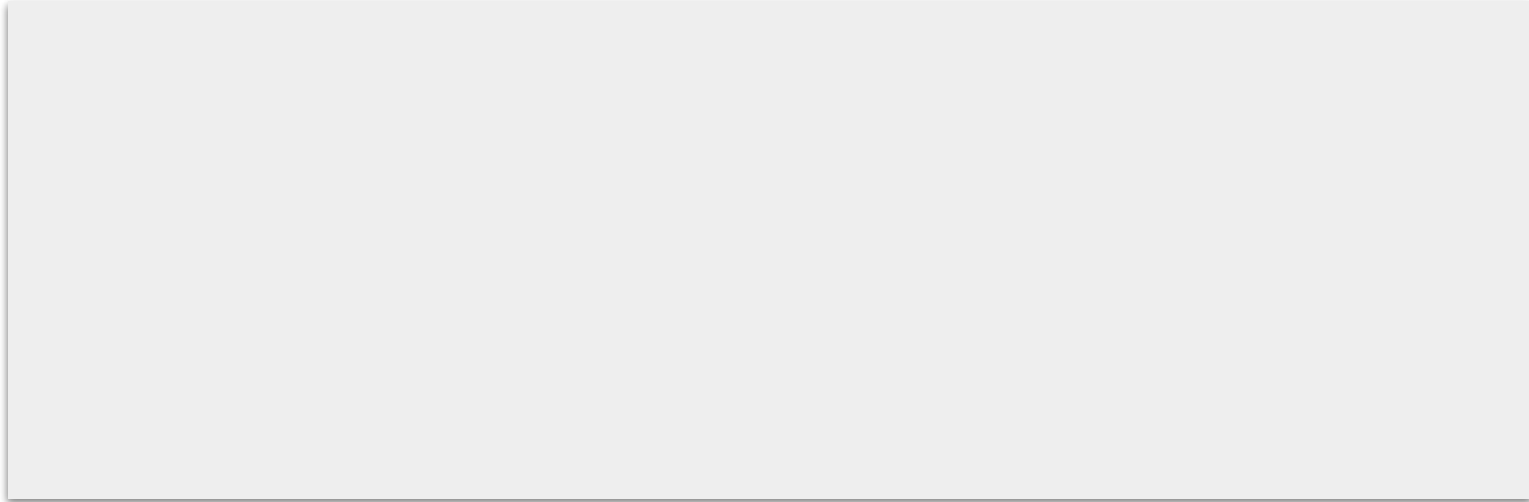
(Dynamischer) Content

-> Laden/Netzwerk



Angular App-Shell

Generiert via Server-Side-Rendering



Angular App-Shell

Generiert via Server-Side-Rendering



Benötigt Angular-Routing

`RouterModule.forRoot()` bzw.
`AppRoutingModule`

Angular App-Shell

App-Shell per Server-Side-Rendering:

```
ng generate app-shell \
  --client-project pwa-demo-ng \
  --universal-project server-app
```

@angular/universal



App-Shell bauen

```
ng run sensor-pwa:app-shell -c production
```

Zusammenfassung

PWA Einstieg

- Fokus auf Nutzer legen
- Features mit höchstem Nutzen zuerst
- SPA + PWA ~ App

Deployment

- Statische Webseite
- Server Side rendered Anwendung
- Single Page Applications
- SPA als mobile App paketieren

HTTPS

*Nutzung von
nativen APIs*

pwabuilder.com

Vielen Dank!

Fragen?

