buschmais

Beratung . Technologie . Innovation

14. Dezember 2017, Dresden

96. JUG Saxony Veranstaltung

Mobile Anwendungen mit Angular und Ionic

buschmais

Beratung . Technologie . Innovation

Frank Schwarz

Teil 2: Progressive Web Apps, Reactive Extensions for Angular

buschmais

Beratung . Technologie . Innovation

Über buschmais

Wir sind ein Dresdner IT-Beratungsunternehmen, gegründet im Jahre 2008. Unsere Schwerpunkte liegen in den Bereichen Softwareentwicklung, Architekturberatung und Training. Gemeinsam mit unseren Kunden entwerfen und implementieren wir fachlich passgenaue Lösungen auf Basis der Java-Technologien.

buschmais GbR

Inhaber
Torsten Busch, Frank Schwarz,
Dirk Mahler und Tobias Israel

Adresse
Leipziger Str. 93
01127 Dresden
info@buschmais.com
http://www.buschmais.de/

universal

isomorphic

responsive

single-page

native

progressive... what?

accelerated

hybrid

web-assembly

mobile

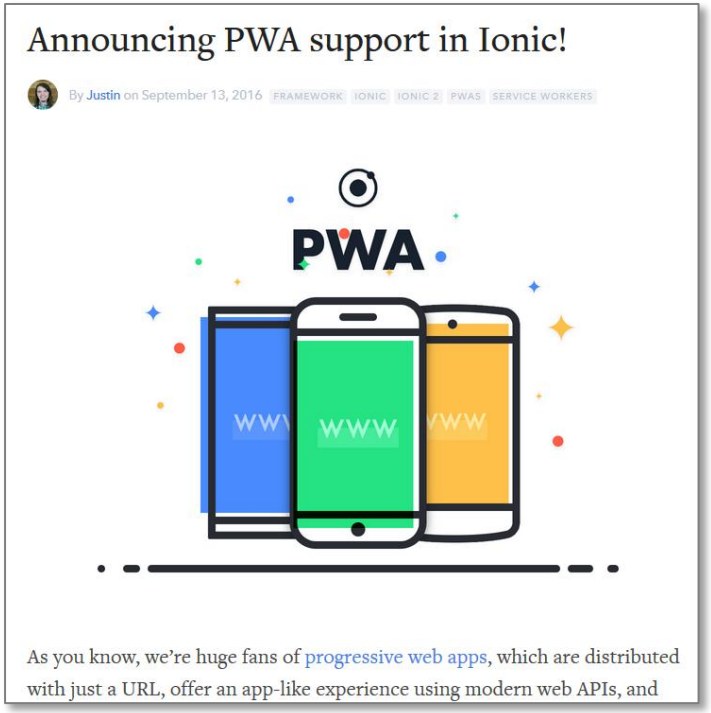
wireless

reactive

Progressive Web Apps
Warum Ionic?

… oder gibt es auch Alternativen?

Phonon

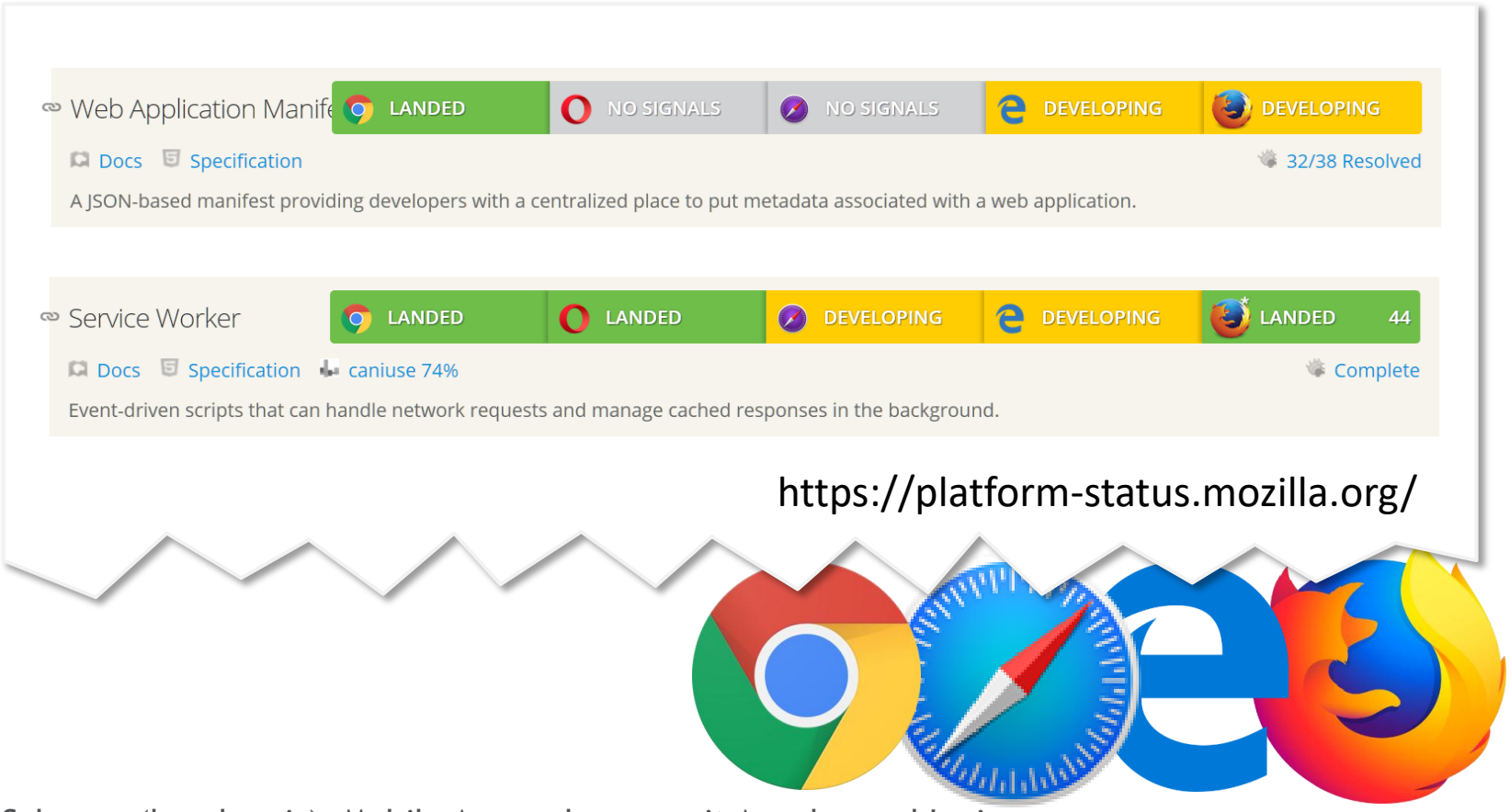



```
$ ionic cordova platform add browser
```

Progressive Web Apps
Worauf kommt es wirklich an?

- Web-App-Manifest: **manifest.json**
  - Name und Logo der App
- Benutzung von **HTTPS**
- Registrierung eines **Service-Worker**s
  - Caching
  - Push-Notifications
  - Offline-Synchronisierung
  - Task-Scheduling
- (Optional): **Meta-Attribute** in der index.html
  - viewport
  - apple-mobile-web-app-title
  - msapplication-tooltip

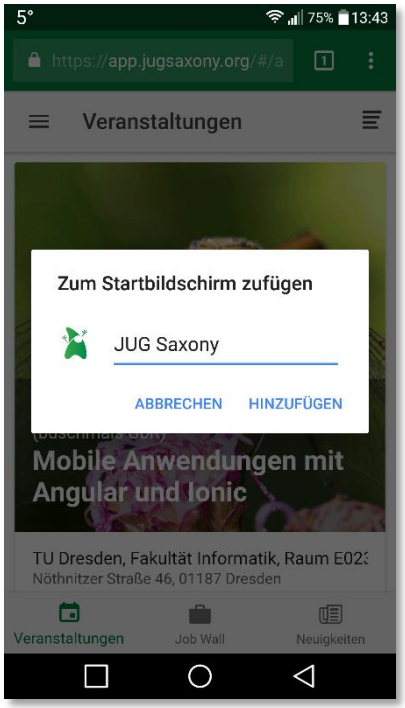
Progressive Web Apps
Worauf kommt es wirklich an?



- Angaben in der **index.html**
  - „traditionelle“ Einstellungen: FavIcon, Title, Theme-Color
  - Viewport-Einstellungen (Zoom, Skalierung, Vollbild-Verhalten)
    ```
    <meta name="viewport" content="...">
    ```
  - Add-To-Homescreen-Einstellungen
    - Standardisiert: application-name
    - Safari: apple-mobile-web-app-title, apple-touch-icon, …
    - Edge: msapplication-TileColor, msapplication-TileImage
  - Referenz auf die „manifest.json“
    ```
    <link rel="manifest" href="manifest.json">
    ```
  - Registrierung des Service-Workers
    ```
    <script>
        if ('serviceWorker' in navigator) {
          // …
    ```

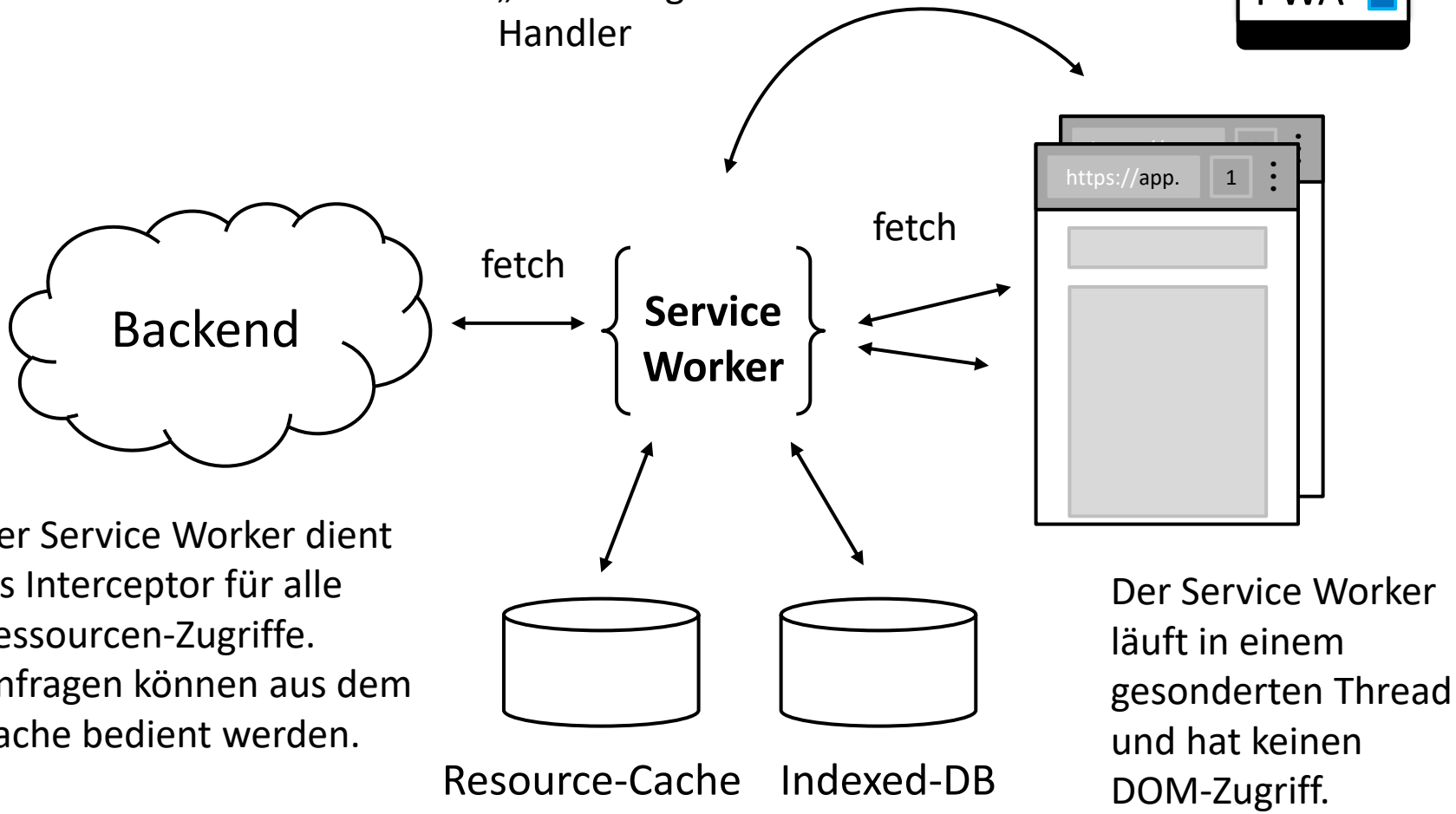
Progressive Web Apps

- Das Web-App-Manifest: **manifest.json**
  - Die Idee ist, den META-Tags aus der index.html einen sinnvollen Platz zu geben
    - Name der Anwendung (lokalisierbar)
    - Icons
    - Beschreibung
    - Start-URL
    - Bevorzugter Anzeigemodus
  - Das Manifest wird derzeit im Wesentlichen dazu genutzt, das App-Icon auf dem Home-Screen zu definieren und den Splash-Screen beim Start der Web-App zu generieren.
  - https://developer.mozilla.org/de/docs/Web/Manifest



Progressive Web Apps
Service-Worker

Datenaustausch via „postMessage()“ und „onmessage“-Event-Handler

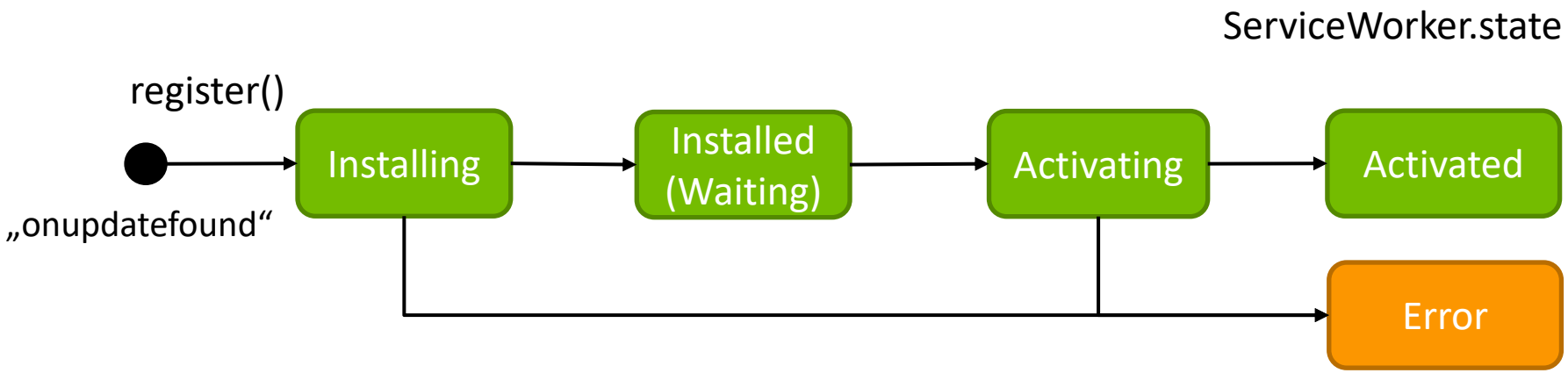


Der Service Worker dient als Interceptor für alle Ressourcen-Zugriffe. Anfragen können aus dem Cache bedient werden.

Der Service Worker läuft in einem gesonderten Thread und hat keinen DOM-Zugriff.

- Die Service-Worker Registrierung (index.html)

```
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('service-worker.js')
      .then(registration => console.log('Service worker registered', registration))
      .catch(error => console.error('Error registering service worker', error));
  }
</script>
```



- Offline-Handling / Caching

```
'use strict';
importScripts('./build/sw-toolbox.js');

self.toolbox.options.cache = {
  name: 'ionic-cache'
};

// pre-cache our key assets
self.toolbox.precache([
  './build/main.js‘,
  './build/vendor.js‘,
  './build/main.css‘,
  './build/polyfills.js‘,
  'index.html‘,
  'manifest.json']);

self.toolbox.router.get('/assets/*', self.toolbox.cacheFirst);
self.toolbox.router.get('/build/*', self.toolbox.cacheFirst);

self.toolbox.router.default = self.toolbox.networkFirst;
```

„sw-toolbox.js“ ist ein Framework zur deklarativen Steuerung des Cache-Verhaltens.

Alternativ:

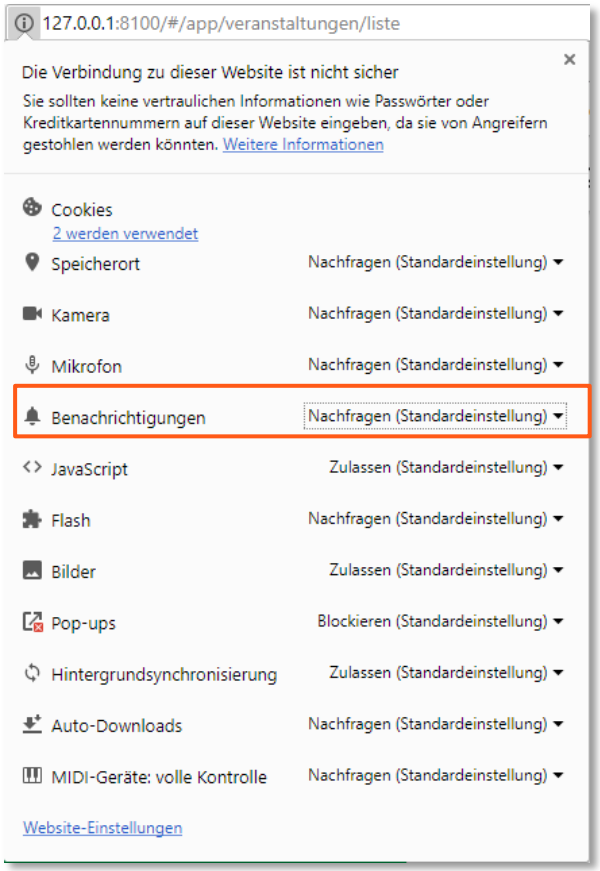
```
self.addEventListener('fetch',
  event => event.respondWith(
    fetch('http://some.url/i.png')
  )
);
```

Jake Archibald: The offline cookbook

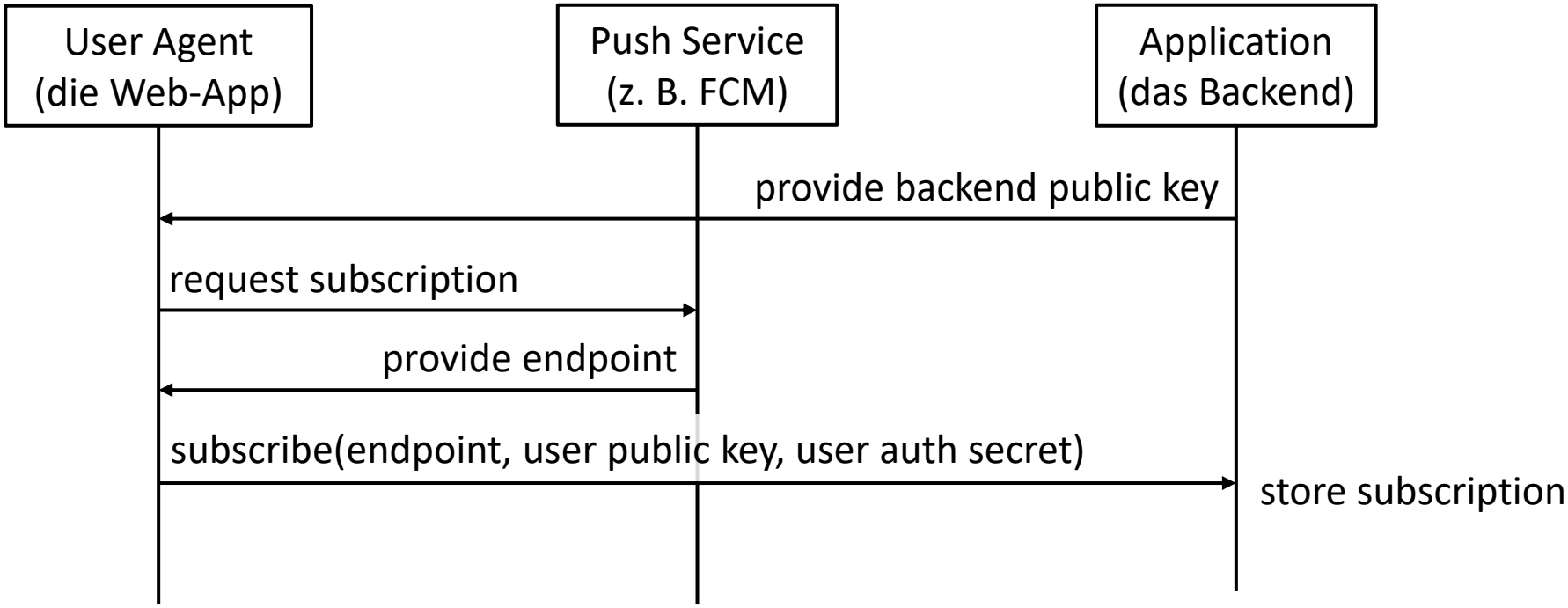
Progressive Web Apps
Push-Notifications

- Notifications / Benachrichtigungen
  - Nachrichten, die vom Betriebssystem (Windows-Tray, Android Notification Center, iOS Notification Center, …) angezeigt werden.
  - Apps, die Notifications erzeugen, brauchen i.d.R. eine besondere Berechtigung.

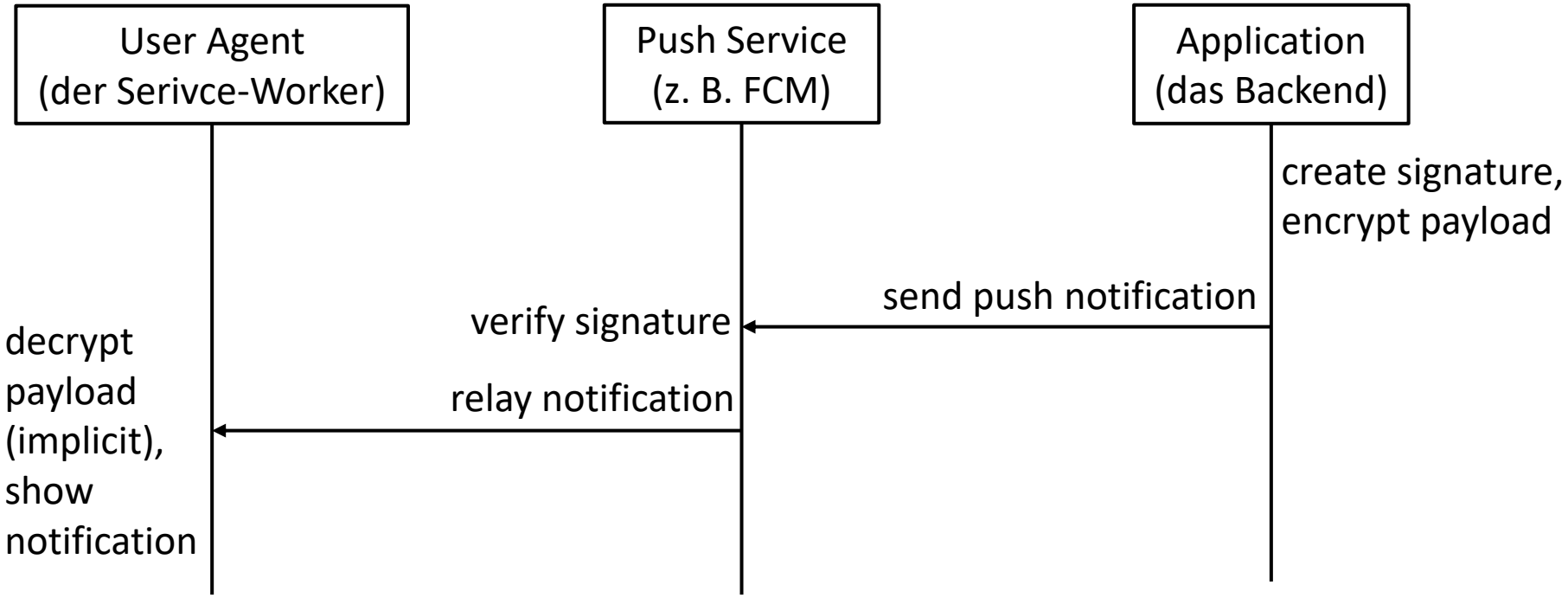
- Push-Notifications
  - Durch ein Backend erzeugte Nachrichten, die von einem Android/iOS-Gerät ohne Zutun einer App zeitnah empfangen werden.
  - Es gibt „silent notifications“, die im Hintergrund verarbeitet werden und vom Nutzer nicht wahrgenommen werden können.



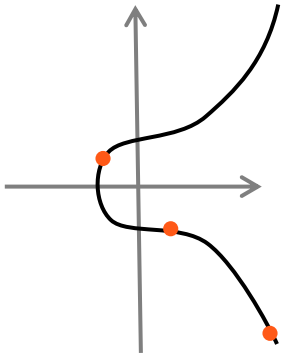
- Das Web-Push Protokoll
  (Push API, https://w3c.github.io/push-api/)



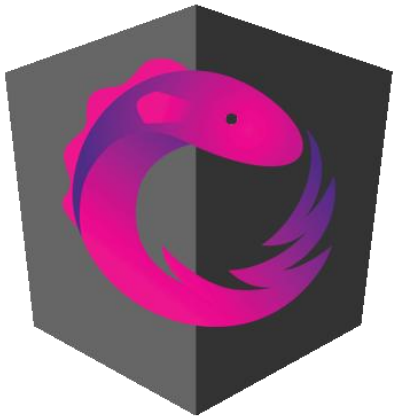
- Das Web-Push Protokoll



- Das Web-Push Protokoll (npm install web-push)
  - Die Nachricht, die das Backend verschickt, muss verschlüsselt und signiert werden:
    - Signatur via JsonWebSignature (RFC 7515)
    - Verschlüsselung via AES-GCM (symmetrische Verschlüsselung)
  - Der symmetrische Schlüssel wird aus dem User-Public-Key, dem Server-Public-Key und dem Auth-Secret des Nutzers abgeleitet (via HKDF-Key-Generator mit SHA256-Digest).
  - Die asymmetrischen Schlüssel basieren auf dem · Elliptic-curve Diffie-Hellman (ECDH) Schlüsselaustausch-Protokoll
  - Für die Generierung des symmetrischen Schlüssels wird ein weiteres ECDH-Schlüsselpaar erzeugt. Der Public-Key wird der Push-Notification via HTTP-Header mitgegeben.

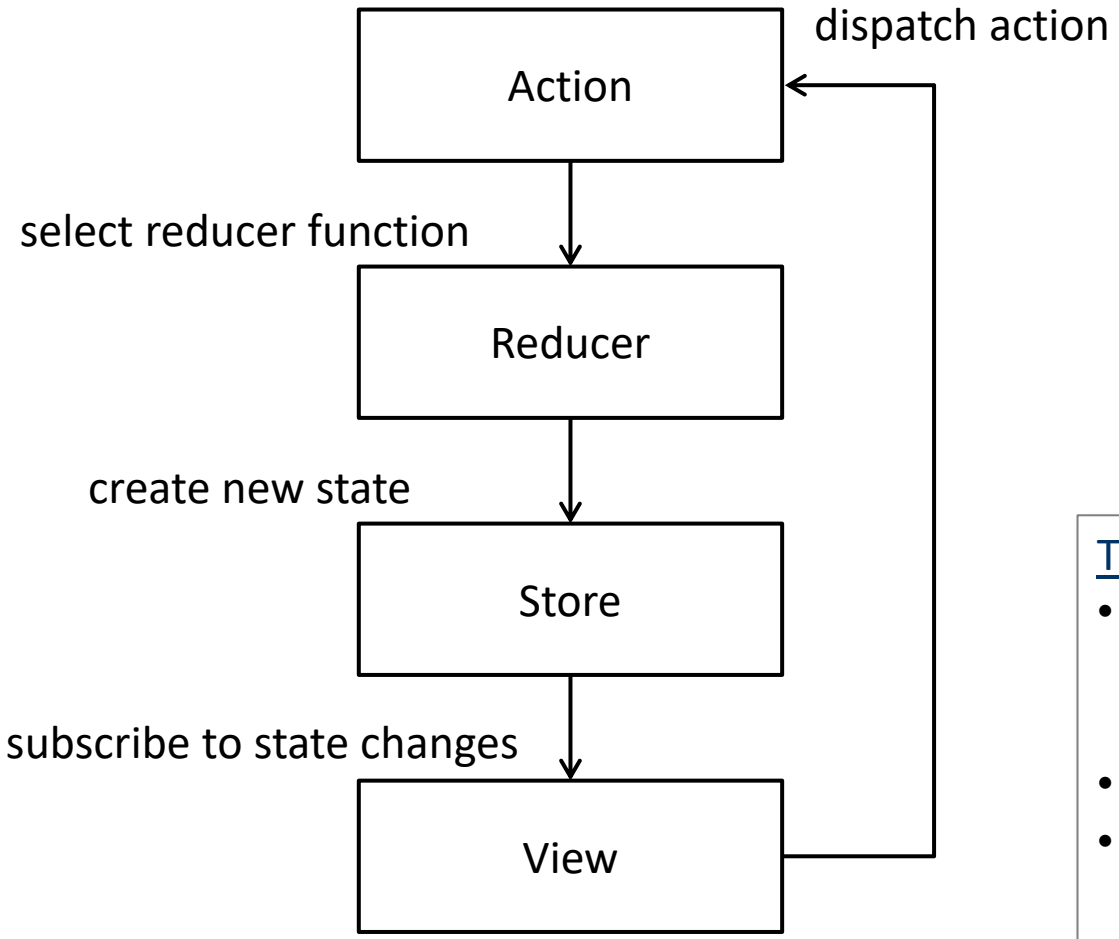


Reactive Extensions for Angular



ngrx

Reactive Extensions for Angular
Der Redux-Ansatz



The three principles of Redux:

- Der Zustand der gesamten Anwendung wird als einzelner Baum verwaltet (→ „Store“)
- Der Zustand ist read-only
- Der Zustand wird durch seiteneffekt-freie Funktionen transformiert (→ „Reducers“)

Reactive Extensions for Angular
Was sind „Reactive Extensions“ (Rx)

- AngularJS ($http Service)
  - ```
    $http
        .get('/someUrl', config)        // liefert eine Promise
        .then(transformingFunction)
        .then(consumingFunction)
        .catch(errorHandlingFunction);
    ```
- Angular (HttpClient)
  - ```
    this.http
        .get('/someUrl', options)       // liefert ein Rx.Observable
        .timeout(someInt)
        .retry(someInt)
        .do(sideEffectFunction)
        .map(transformingFunction)
        .catch(errorHandlingFunction)
        .subscribe(consumingFunction)   // liefert eine Rx.Subscription
    ```

Reactive Extensions for Angular
Pattern und Antipattern

- Konsum eines Observables in einer Komponente

```
refesh(): void {
  this.eventService.getEvents()
    .subscribe(events => this.events = events);
}
```

normale Variable (Event[])

- Alternativ: Konsum eines Observables im Template

```
this.events$ = this.eventService.getEvents();

<event-card
  *ngFor="let event of events$ | async"
  [event]="event">
</jug-event-card>
```

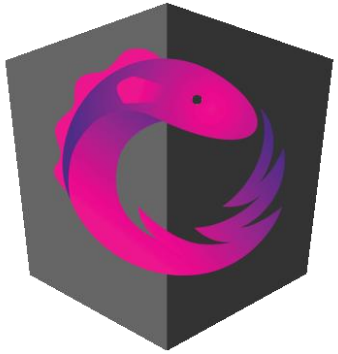
Observable<Event[]>

Reactive Extensions for Angular
Pattern und Antipattern

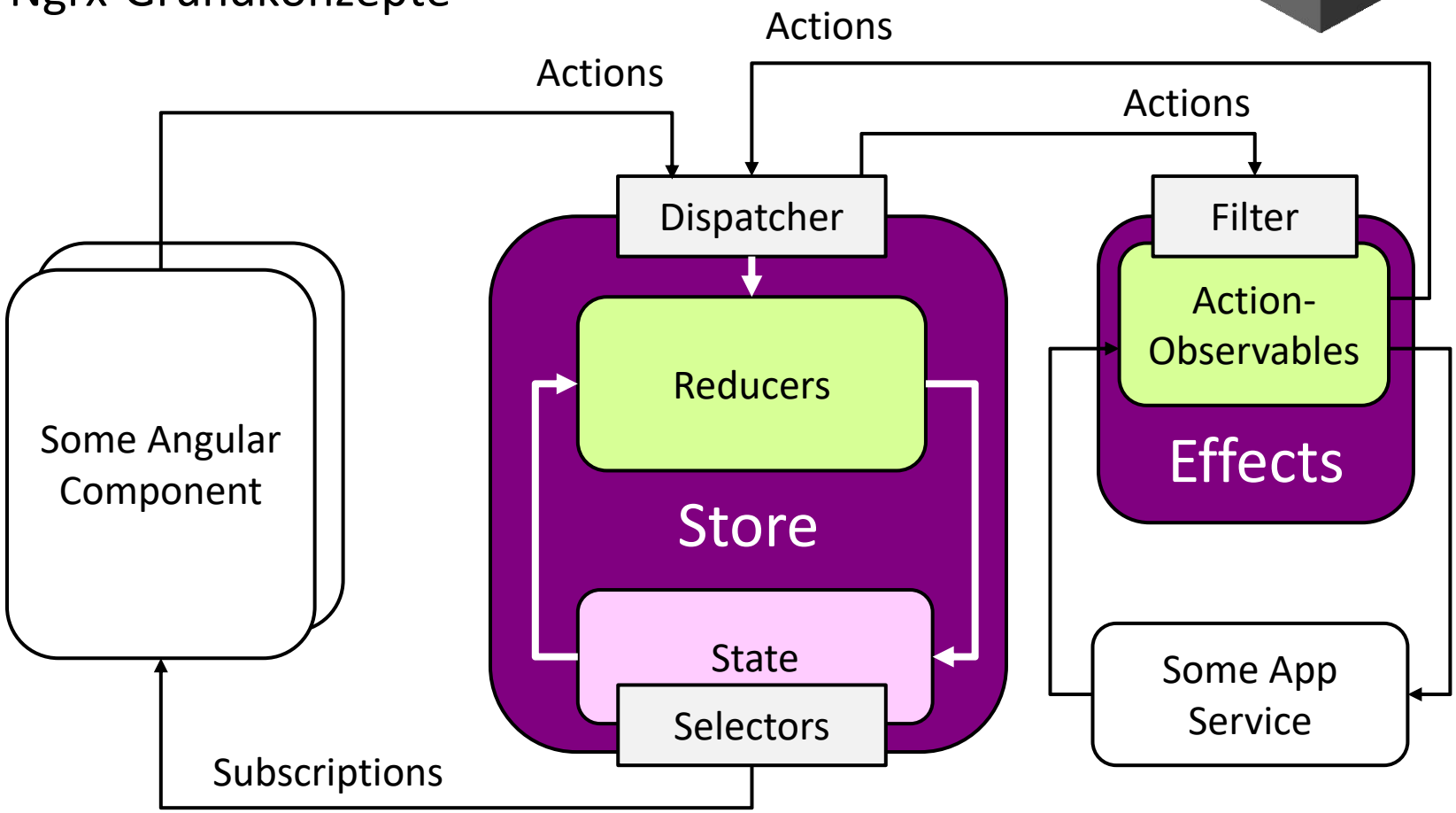
- Probleme im Umgang mit Observables
  - Ohne Subscription wird keine Aktion getriggert.
  - Jede neue Subscription triggert die gesamte Kette.
    - → HTTP-Anfragen werden mehrfach ausgeführt.
    - Gegenmittel:
      - Das Ergebnis als Zustand in der Komponente speichern.
      - Das Ergebnis in der View „speichern“ (`*ngIf="event$ | async as event"`)
      - Das Observable teilen (`Observable#share()`) → Hot-Observable
  - Hot-Observables erzeugen Memory-Leaks, wenn kein „unsubscribe()“ stattfindet.

- Probleme im Umgang mit Observables
  - „Higher-order“ Observables sind kompliziert

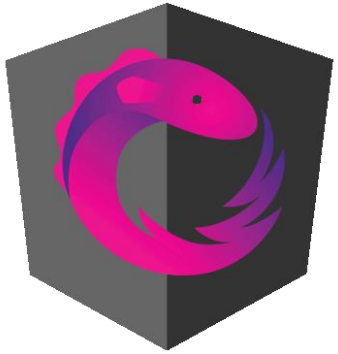
```
public events$: Observable<Event[]>;

private poller$: Subject<void> = new Subject();

constructor(eventService: EventService) {
  this.events$ = eventService.getEvents()
    .repeatWhen(notifier => notifier.flatMap(_ => this.poller$));
}

repeat(): void {
  this.poller$.next();
}
```



- Ngrx-Grundkonzepte

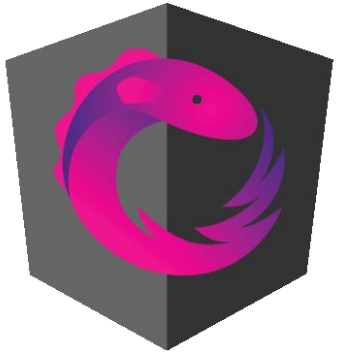


Reactive Extensions for Angular
Grundkonzepte von ngrx



Effects

```
import * as eventListActions from '../actions/event-list.actions';

@Injectable()
export class EventListEffects {

    constructor(private actions$: Actions, private eventsService: EventsService) { }

    @Effect()
    refresh$: Observable<Action> = this.actions$
        .ofType<eventListActions.Refresh>(eventListActions.REFRESH)
        .concatMap(action =>
            this.eventsService
                .getEvents()
                .timeout(10000)
                .map(list => new eventListActions.RefreshSuccess(list))
                .catch(error => of(new eventListActions.RefreshFail(error)))
        );
}
```



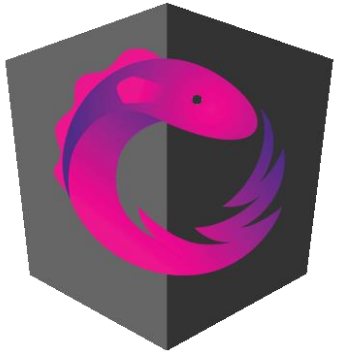
Actions

```
export const REFRESH = '[EventList] Refresh';
export const REFRESH_SUCCESS = '[EventList] Refresh Success';
export const REFRESH_FAIL = '[EventList] Refresh Fail';

export class Refresh implements Action {
    readonly type = REFRESH;
}

export class RefreshSuccess implements Action {
    readonly type = REFRESH_SUCCESS;
    constructor(public payload: EventEntity[]) { }
}

export class RefreshFail implements Action {
    readonly type = REFRESH_FAIL;
    constructor(public payload: any) { }
}

export type Actions = Refresh | RefreshSuccess | RefreshFail;
```

Folgt man diesem Muster, kann die Fehlerbehandlung nicht mehr vergessen werden.

Im mobilen Umfeld sind Netzwerkfehler nicht die Ausnahme, sondern der Normalfall.

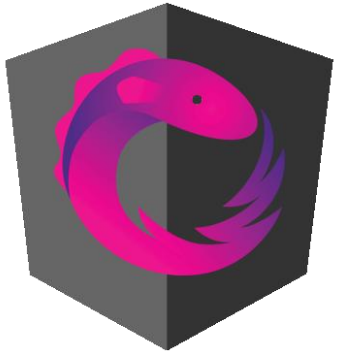
Reactive Extensions for Angular
Grundkonzepte von ngrx



- Reducers und State

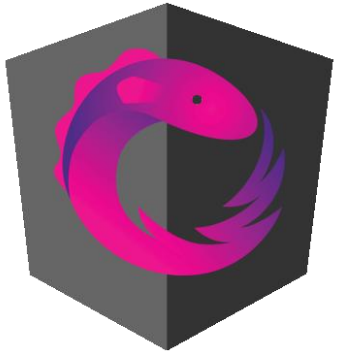
```
export interface State {
    events: EventEntity[];
}

export const initialState: State = {
    events: []
}

export function reducer(state = initialState, action: eventList.Actions): State {
    switch (action.type) {
        case eventList.REFRESH_SUCCESS:
            return {
                ...state,
                events: action.payload
            };
        default:
            return state;
    }
}
```

Reactive Extensions for Angular
Grundkonzepte von ngrx



- Selectors

```
export const getEntities = (state: State) => state.entities;


export const getEventsState = createFeatureSelector<fromApp.State>('events');

export const getEventsEntitiesState = createSelector(getEventsState, getEntities);
```

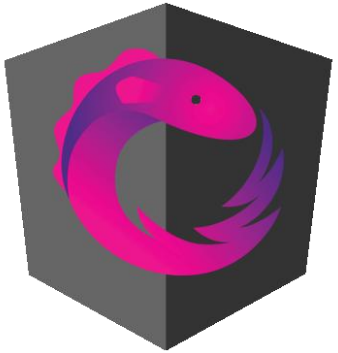


Anbindung an die Angular-Anwendung

```
@Component({
  selector: 'page-event-list',
  templateUrl: 'event-list.page.html',
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class EventListPage {

  public entities$: Observable<EventEntity[]>;

  constructor(private store: Store<State>) {
      this.entities$ = store.select(getEventsEntitiesState);
  }

  ionViewDidEnter(): void {
    this.store.dispatch(new eventListActions.Refresh());
  }
}
```

Wir können die Change-Detection von Angular deaktivieren. Die PWA läuft dadurch wesentlich ressourcen-schonender.

Progressive Web Apps
Resümee



- Ist Ionic (mit Angular) tauglich für PWAs?
- Sind Progressive Web Apps bereit für die große Bühne?
- Bringt „ngrx“ Vorteile für die Entwicklung einer PWA / Angular-Anwendung?

buschmais

Beratung . Technologie . Innovation

Vielen Dank. Fragen bitte an:

Frank Schwarz
buschmais GbR

Inhaber
Torsten Busch, Frank Schwarz,
Dirk Mahler und Tobias Israel

frank.schwarz@buschmais.com
http://www.buschmais.de/

Dresden, 14. Dezember 2017