

# ng2react



## A React that eats an Angular:

Angular to React migration in ten lines of code



Schreibe einen Beitrag

Teile Neuigkeiten und Beobachtungen



**SCHNAPPSCHUSS**

Das ist die erste deutsche Computermouse, die parallel zur ersten amerikanischen Mouse entwickelt wurde. Die "Rollkugel" wird von ihrem Entwickler, Rainer Mallebrein, gehalten und ist im Heinz Nixdorf...



Kristina Grube (Neue Westfälisch...

vor 4 Stunden • Fürstenallee 7, Pade...



2



0 Kommentare



Teilen



SEITE

## "Womo Ausfluggipp: Paderborn" bei YouTube

Paderborn ist auch für Reisemobilisten ein lohnenswertes Ziel. Das veranschaulicht das Video des Kanals GI Womo News, welches den attraktiven Stellplatz am Rolandsweg und die Sehenswürdigkeiten...



Tourist Information Paderborn

vor 2 Stunden • Marienplatz 2a, Pad...



Mag ich



0 Kommentare



Teilen

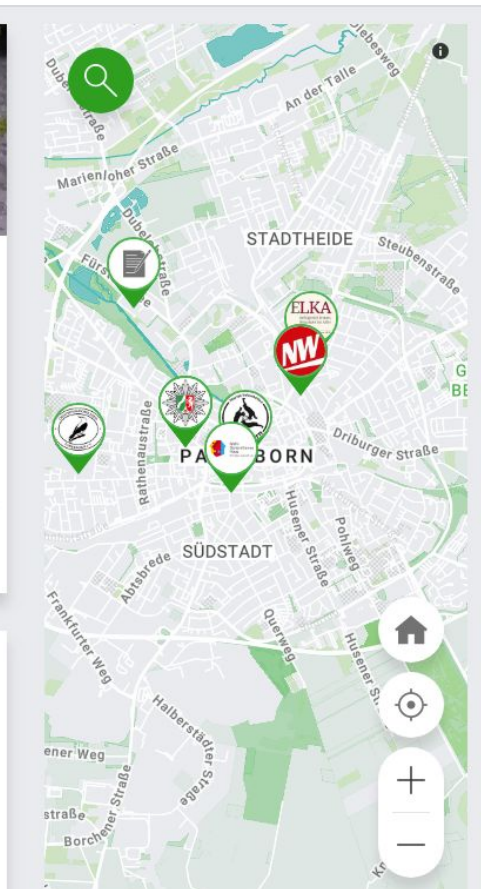
ÖFFENTLICH

## POL-PB: Gänseküken bei nächtlichem Feuer verendet

Delbrück (ots) - (mb) Ein Gänsestall mit bis zu 6.000 Gänseküken ist in der Nacht zu Dienstag durch ein Feuer zerstört worden. Das Feuer an der Steinhorster Straße wurde kurz nach 03.00 Uhr von



Meldungen der Polizei Paderborn



lokalportal

# Lior Oren

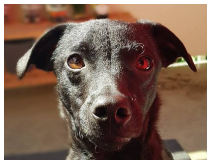


Head of Frontend in Lokalportal



7 years in HH <3

From Tel-Aviv



Dogs

Metacafe, Wix,  
Jimdo, Smaato



@alodium

History



# Lokalportal Stack history

---

**2014:**

PHP

Angular 1.x

**2017:**

couldn't move anymore

What do you do when you cannot scale your app anymore?

**Refactore/migrate**

# Migration concerns - Tech

---

- We want the newest tech in the industry (not github)
  - Align with the industry
  - Align with the current browsers, standards
  - Attract developers - easy to find devs in a mainstream tech
  - Easy onboarding for new developers
  - Solutions over the internet

So we chose  React

# Business Migration concerns

---

- We can't build a dead app for a year
  - Code not in prod is a worthless code:
    - The company can't earn, users can't enjoy
    - We don't really know if the system works
    - Real users - Real bugs
  - We can't really afford ourselves to have two apps - maintain, resources from other teams (design, backend - all need to support two apps).
  - New requirements:

By the time we finish a featureset, the company's roadmap and scope might change

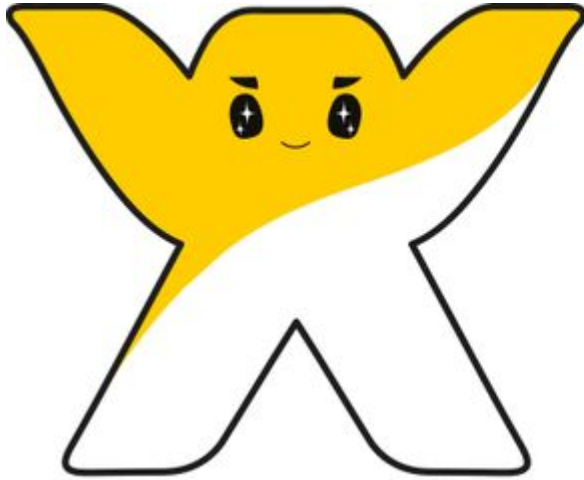
# Migration Rule 0: Have everybody onboard

---

All developers should agree on the direction

- Some developers won't like being out of the comfort zone of the old stack, afraid of learning new stuff.
- Some are into other stacks (don't be in love with one technology)
- Other teams needs to know to support you, and the limitations of your new stack

# Sleeping App:



**wix**



# Sleeping App:

It's bad. Non-prod-code is bad code!



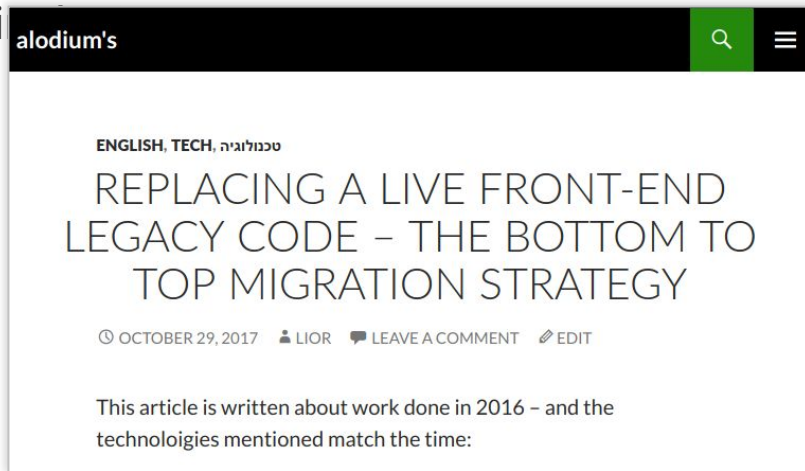
So we chose Bottom-up migration

# 2016 The primefaces experience



Another company had a Primefaces frontend - auto generated frontend with jquery. The devs just built components with Java + hacks i

To be able to build fancy frontend features, we migrated to Angular (1.x)



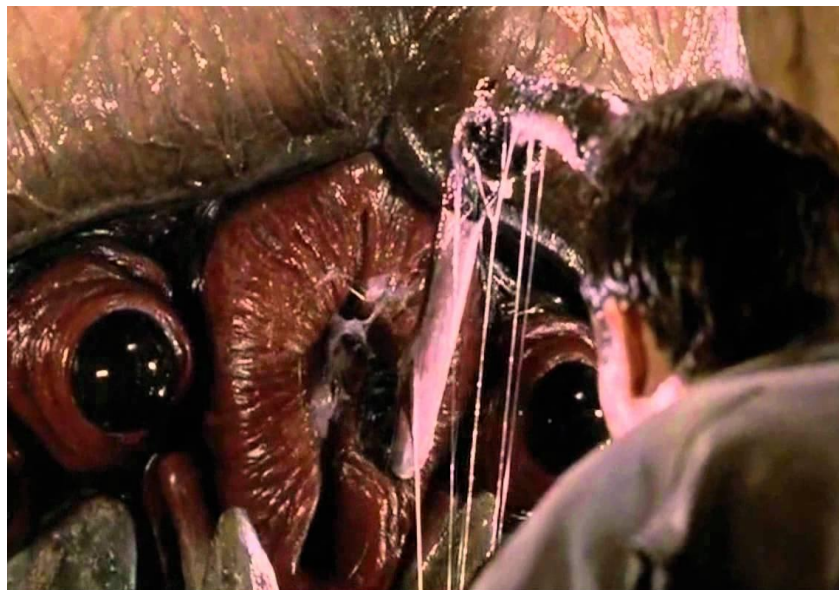
# Bottom-up approach

---



Allow your new app to eat the old app  
from bottom to top, grow from within

- Each feature is immediately in production
- No waste of money
- Stay close to the feature scope,  
business logic
- Immediate feedback from your users



And it's more fun ^^

# Bottom-up approach

How?



# Migration Rule 1: Isolation, encapsulation

---

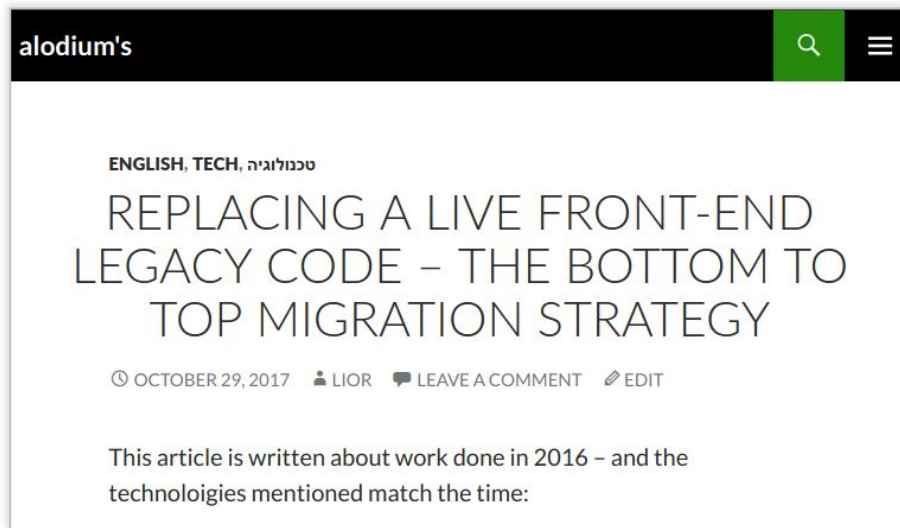
**Isolate your two apps so they won't be able to interrupt with each other.**

- Isolate dom nodes - don't let one app to listen or to change the other's dom
- Check who is using and overriding window objects
- Libraries (jquery used? JQuery version? Lodash?)

# How: The primefaces experience



Render an Angular app in the static HTML, that is not controlled with Primefaces.



# Lokalportal migration - Given

---

- The Angular app was in an unextendable point.
- We knew that we want React, we knew that we do bottom to top.
- Solutions on the internet are heavy, not allow code separation. Most asked you to write react code within Angular components.
- For example [angular-react/fabric](#)

# Oh, and

the angular was written with  *CoffeeScript*

---

## We love Typescript <3 <3



# Migration Rule 2: seperation

---

We wanted a separate, new app.

You take the folder out and it works! No two apps mixed.

- App (Angular)
- React-portal
- Admin
- Storybook
- ...

# Migration Rule 3: Single point of truth

---

- Have only one data source, for both apps
  - Expose api between the apps with the holy `window` object (example will follow)
  - Allow only one communication layer, with this data source

# Experiment day

---

- We splitted into two teams, one tried to render the Angular components inside react app
- The second - a react component inside angular



React is so componental...  
We had a winner (and one weird Angular in React app)

# NG2REACT - Code Examples



# NG2REACT - Exposed React components

[File on gitlab](#)

```
// Use a map to contain a list of allowed components
const components: IComponentsMap = {
  eventFeedContainer: EventFeedContainer,
  headerContainer: HeaderContainer,
  landingPageContainer: LandingPageContainer,
  mailboxContainer: MailboxContainer,
  mapContainer: MapContainer,
  metaTags: MetaTags,
  newsFeedContainer: NewsFeedContainer,
  postsDetailsContainer: PostsDetailsContainer,
  profileFeedContainer: ProfileFeedContainer,
  settingsMessageContainer: SettingsMessageContainer,
  settingsPageContainer: SettingsPageContainer,
};
```

# NG2REACT - The “magic”

[File on gitlab](#)

```
w.lopo_app.ng2react = (component: string,  
                        el: HTMLElement,  
                        props: object,  
                        store?: Store<IStore>): Element => {
```

- Component - list of allowed components (Couldn't type this /: )
- El - Angular dom element (directive)
- Props
- Store (optional) - if it's a container

# NG2REACT - (Rule 1: Isolate)

```
const baseComponent = <MyComponent {...props} ng-non-bindable='true' />;
```


Don't let Angular touch the inside dom of React

# NG2REACT - The store

If there is a store passed - wrap it with a provider:

```
const baseComponent = <MyComponent {...props} ng-non-bindable='true' />;
```

```
return ReactDOM.render(  
  getProvider(baseComponent, store),  
  el,  
) as Element;
```



*If a store is injected, wrap the component with a store provider*

```
function getProvider(component: JSX.Element, store?: Store<any>): JSX.Element {  
  if (store) {  
    return (  
      <Provider store={store}>  
        {component}  
      </Provider>  
    );  
  }  
  return component;  
}
```



# NG2REACT - The “magic”

```
w.lopo_app.ng2react = (component: string,  
                        el: HTMLElement,  
                        props: object,  
                        store?: Store<IStore>): Element => {  
  const MyComponent = components[component];  
  // Generic creation of the component  
  // ng-non-bindable='true' = this will cause angular to ignore that HTML piece.  
  const baseComponent = <MyComponent {...props} ng-non-bindable='true'/>;  
  
  return ReactDOM.render(  
    getProvider(baseComponent, store),  
    el,  
  )as Element;  
};
```

# NG2REACT - The Angular

react\_feed\_profile.directive.coffee 515 Bytes



```
1 module.exports = (Redux, $stateParams) ->
2   replace: true
3   template: '''
4     <div class="react-profile">
5     </div>
6     '''
7
8   link: (scope, element) ->
9     Redux.getStore().then (store) ->
10       container = element[0]
11
12       window.lopo_app.ng2react('profileFeed', container, {
13         profileId: $stateParams.identifier.toLowerCase()
14         profileType: $stateParams.profileType.toLowerCase() + '_profile'
15       }, store)
16
17       scope.$on '$destroy', ->
18         window.lopo_app.unmountReactComponentAt(container)
```

# NG2REACT - (Rule 3: Single point of truth!)

## The store:

Expose the store for Angular

```
// Once the redux store is ready - Expose it to Angular  
export function exposeStore(store: Store<IStore>) {  
  w.lopo_app.store = store;  
  w.lopo_app.resolveStoreCreation(store);  
}
```

That's it! (for the technical part)

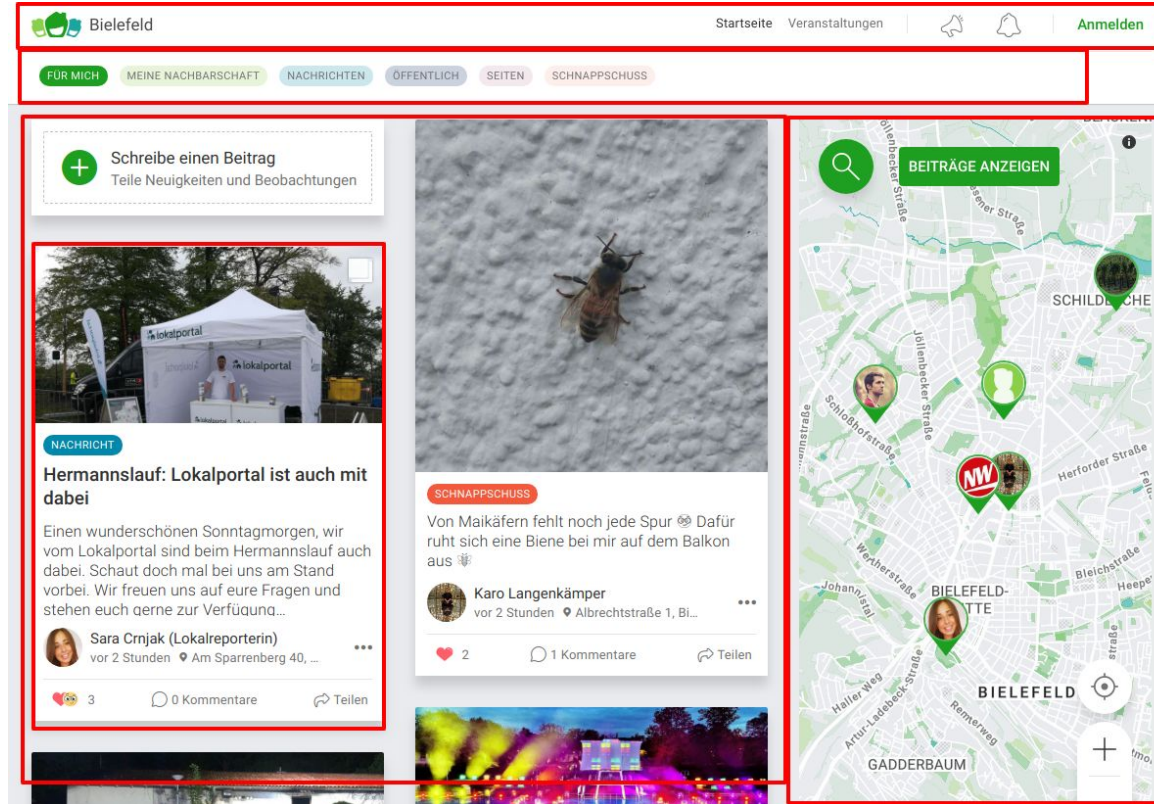
# Questions?

# Flow

---

- We don't change existing angular component - if it needs a change, we build new react component
- New features are built in React
- Before migrating a feature we re-think if it is needed in the first version

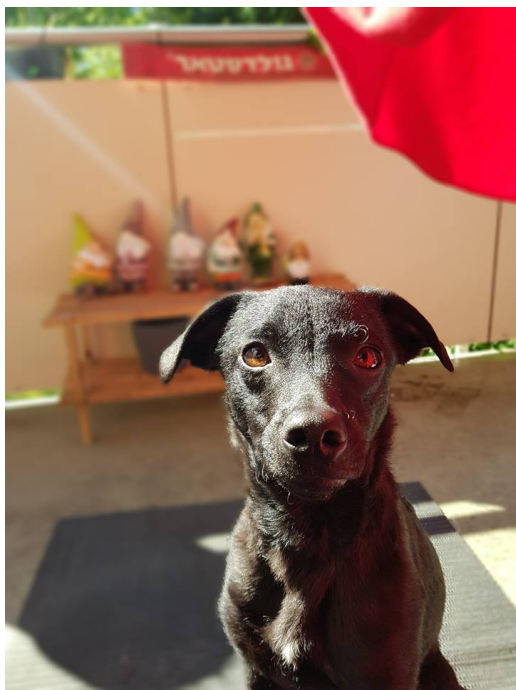
# Slow eating of the app



# Last and hardest - the router

---

- Angular ui router
- The last “component”
- Many views had no route, just actions that rendered another view
- When finally changing to React Router - we were smarter to know about what we want from our routes.



**Done**  
1 year.