CONCEPTS OF REACT
Johannes Reuter

Master student HS Karlsruhe

&

Student employee @ inovex lab

evaluation of new (web-)technologies

jreuter@inovex.de
React Facts

- Library (NOT a framework)
- Is only the view-part of the application
- Open source
- Developed (and used) by facebook
Coming from layers...

Structure (HTML)

Appearance (CSS)

Behavior (JS)
...to components

Component

JS + JSX

Component

JS + JSX

Component

JS + JSX

API

API

API

separation of concerns
Hierarchical structuring
App state vs. UI-state

DATA a.k.a. app state

UI-state
On new data

- Complete re-evaluation
- Comparison of result to displayed components
- Calculation of minimal set of changes
  - JS (fast)
- Execution of changes
  - DOM (slow)
How to define Components?
Example Todo-App

What needs to be done?

- Eat
- Pray
- Love

3 items left
0. Data modelling

```json
{
    "todos": [
        {
            "name": "Eat",
            "completed": false
        },
        {
            "name": "Pray",
            "completed": false
        },
        {
            "name": "Love",
            "completed": false
        }
    ]
}
```
1. Separate
2. Define and connect
data selectively passed down
3. Zoom in
4. Repeat
The bigger picture

DATA

how does this part work?
React leverages reactive programming

- Modelling of data-flow
- Data-changes are propagated automatically
- No need to synchronize model <-> view (data-binding)
On application level (Excel)
On code level (RxJS)

```javascript
const subscription = todos
  .filter(todo => !todo.completed)
  .count()
  .subscribe(
    count => console.log(`There are ${count} uncompleted todos`),
    err => console.log(`Something went wrong: ${err.message}`);
  );
```
On architecture level (Angular 1)
Problem: Loops in the flow

Watchers fired in the last 5 iterations: []
http://errors.angularjs.org/1.4.3/$rootScope/infdig?p0=10&p1=%
    at angular.js:68
    ...

450 Questions on Stackoverflow
Something changes something changes something changes something...
Solution: Unidirectional dataflow
App-state is managed in centralized place
Flux

- **Action**
- **Dispatcher**
- **Store**
- **View**

- on async event
- on user input

- **app state**

https://facebook.github.io/flux/docs/overview.html
Alternative (Redux)

http://staltz.com/unidirectional-user-interface-architectures.html
Alternative

Event Stream (Server side)

Action

on user input

View

push

Aggregated Database-View

And much, much more

- Cerebral
  [www.cerebraljs.com/](http://www.cerebraljs.com/)

- NuclearJS
  [optimizely.github.io/nuclear-js/](http://optimizely.github.io/nuclear-js/)

- Alt
  [alt.js.org/](http://alt.js.org/)

- FluxThis
  [www.fluxthis.io/](http://www.fluxthis.io/)

- Microcosm
  [github.com/vigetlabs/microcosm](http://github.com/vigetlabs/microcosm)

- ...
Integration in existing architectures

>
Serverside rendering + glue code

>
BackboneJS

>
...
Serverside rendering + glue code

Server

Templating Engine

HTML

jQuery

$(...).html(response);

Event

DOM

AJAX

Serverside rendering + glue code

Server

json_encode

Server

JSON

React

React.render(...);

AJAX

Event

DOM
BackboneJS

**react.backbone**

- Auto-rerenders on model/collection-change
- Place react components into backbone views
Angular

DON’T

◦ Already has a view-engine in place
◦ Big JS-files even for small apps
◦ Boilerplate-code
Upshot

- Hierarchical components defined by interfaces
- Data is passed selectively top to bottom
- Complete re-evaluation on each action
- Unidirectional data flow
- Centralized place for storage of app-state
Questions?