React Komponenten

Themen heute

- Besprechung Übung 6
- Virtueller DOM
- Typen von React Komponenten

Besprechung Übung 6

- Aufgabe 1: Schritte

  create-react-app react-counter

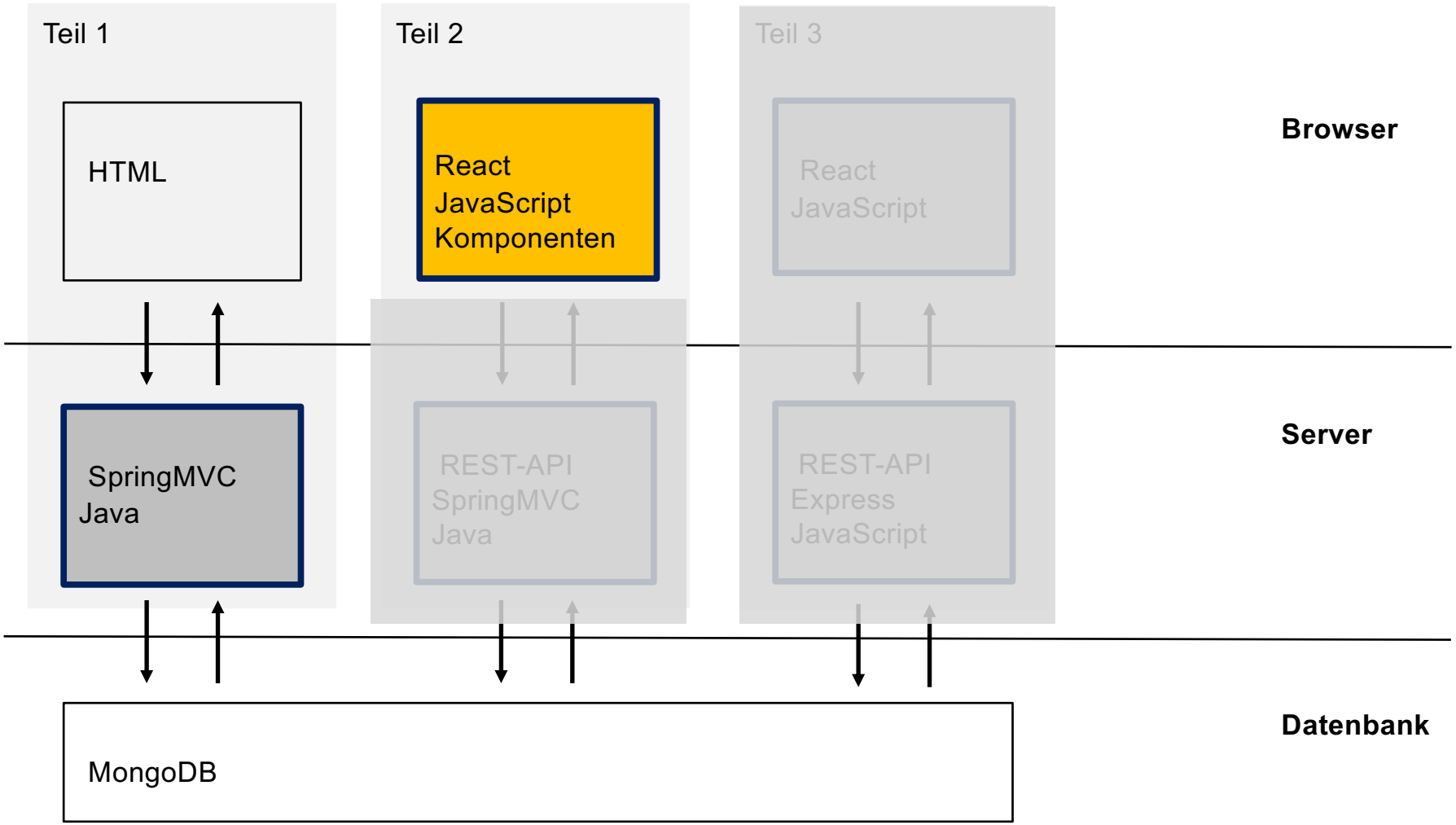
  cd react-counter

  npm install

  npm start

- Aufgabe 2: Code analysieren
  - increment(): State "counter" inkrementieren
    - JavaScript Methode
    - Binding an Instanz im constructor()
  - reset(): State "counter" auf 0 setzen
    - JavaScript Methode
    - Binding an Instanz im constructor()
- Aufgabe 3: Bootstrap einsetzen
  - npm install --save reactstrap bootstrap
  - Im File "index.js" Stylesheet "bootstrap.css" importieren

Lab "flashcard": Setup



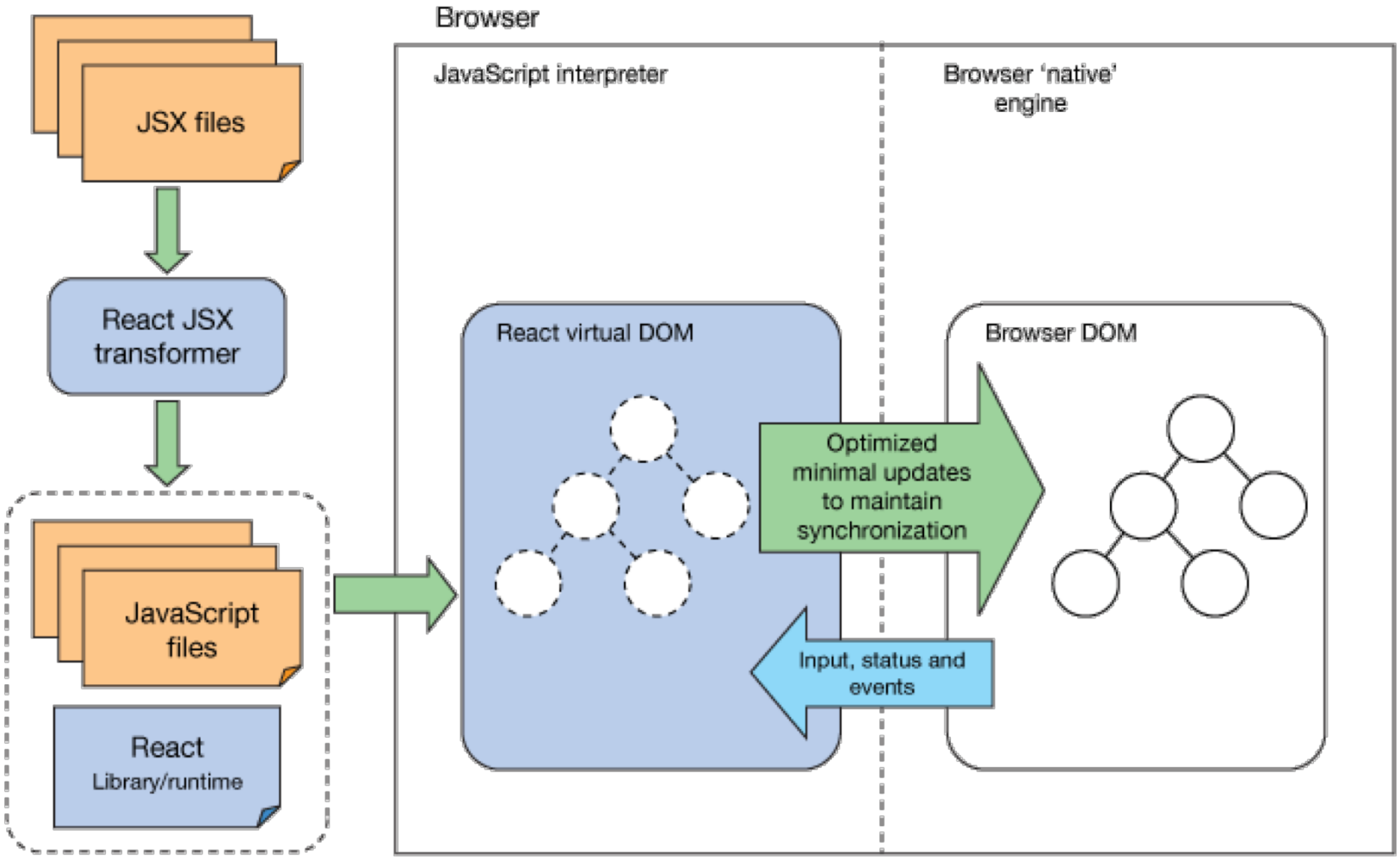
Virtueller DOM (1/3)

- Die Manipulation des DOM (Document Object Model) im Browser ist teuer - und deshalb langsam
  - Jede Änderung an dieser Baumstruktur quittiert der Browser mit teurer Neuberechnung seiner Geometrie.
  - Je mehr geändert wird, desto länger dauert es. Je weniger man den DOM des Browsers verändert, desto schneller ist die Applikation.
- JavaScript selber ist performant
  - Viele JavaScript-Frameworks suchen einen Kompromiss mittels Verwendung von altbewährten Entwurfsmustern, um die Komplexität zwischen Einfachheit und Performance zu bändigen (Two-way binding, dirty-checking, … etc.).
  - React versucht es mit einem extremen Ansatz, der in erster Linie die Einfachheit und nicht die Performanz in den Fokus stellt:
    React rendert bei jedem Update einfach alles neu.

Virtueller DOM (2/3)

- Virtueller DOM
  - Mit React Komponenten und JSX arbeitet man nicht direkt mit dem DOM des Browsers, sondern mit normalen JavaScript Objekten (=> Virtueller DOM), die schnell gelesen und bearbeitet werden können, ohne dass damit tatsächliche Änderungen am DOM ausgelöst werden.
  - Bei jeder Änderung der Daten erstellt React einen neuen virtuellen DOM.
  - Ein stark optimierter und heuristischer Algorithmus vergleicht diesen neuen Baum mit den vorherigen und errechnet eine Liste von minimalen Änderungen am richtigen DOM aus.
  - Diese werden gesammelt und nicht direkt, sondern im Batch an den Browser weitergeleitet.

Virtueller DOM (3/3)



from https://www.ibm.com/developerworks/library/wa-react-intro/

Zusammenfassung React

... für die Implementation von grossen Applikationen
... ist eine JavaScript Bibliothek für den View einer SPA
... reduziert die Applikationsentwicklung auf reines JavaScript
... basiert auf JavaScript-Komponenten
... führt einen Virtuellen DOM
... minimiert dadurch die Manipulationen des DOMs im Browser
... nutzt keine HTML-Templates
... führt JSX als Erweiterung von JavaScript ein
... nutzt Properties um read-only Eigenschaften zu definieren
... nutzt State um den Zustand einer Komponenten zu beschreiben
... nutzt die Methode render() der Komponente, um die Komponente zu zeichnen
... ruft bei jeder Änderung des State die Methode render() automatisch auf

React Komponente: Lifecycle Methoden

- React stellt verschiedenen Lifecycle Methode zur Verfügung, um z.B. Komponente zu initialisieren:

  componentDidMount()

  Die Methode wird einmalig ausgeführt, wenn eine Komponente in den DOM gerendert wird. Aus der React Doc: "If you need to load data from a remote endpoint, this is a good place to instantiate the network request. Setting state in this method will trigger a re-rendering."

  componentWillUnmount()

  Die Methode wird einmalig ausgeführt, bevor die Komponente aus dem DOM entfernt wird. Aus der React Doc: "Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any DOM elements that were created in componentDidMount"

Arbeitsblatt 14

- App "react-counter" um einen Timer erweitern
  - In **componentDidMount()** Timer starten

    ```
    this.timer = setInterval(this.increment, 1000);
    ```

  - In **componentWillUnmount()** Timer stoppen

    ```
    clearInterval(this.timer);
    ```

- Alternativen zum Binding der Event Handler kennenlernen

React Komponente als Functional Component

siehe https://hackernoon.com/react-stateless-functional-components-nine-wins-you-might-have-overlooked-997b0d933dbc

```
import React from 'react';
import { Jumbotron } from 'react-bootstrap';

export default class Header extends React.Component {
  render() {
    return <Jumbotron>
      <h1>{this.props.title}</h1>
      <h3>{this.props.subtitle}</h3>
      </Jumbotron>
  }
}
```

React Component

- Komponente ist eine Klasse
- **Stateful**

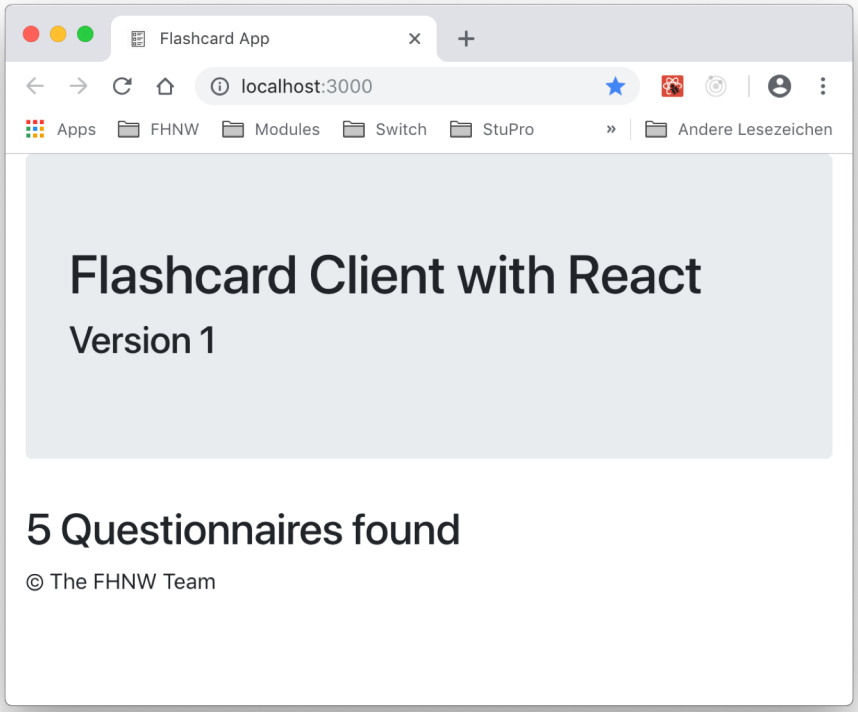
```
import React from 'react';
import { Jumbotron } from 'react-bootstrap';

const Header = ({ title, subtitle }) => (
  <Jumbotron>
    <h1>{title}</h1>
    <h3>{subtitle}</h3>
  </Jumbotron>
)

export default Header;
```

React Functional Component

- Komponente ist eine Funktion
- **Stateless**
- Kein "this" Keyword
- ...

Arbeitsblatt 15

- Initialisierung "Flashcard App" mit Tool "Create React App"
- Komponenten "App", "Header", "Footer" implementieren, Dummy für Komponente "QuestionnaireContainer"

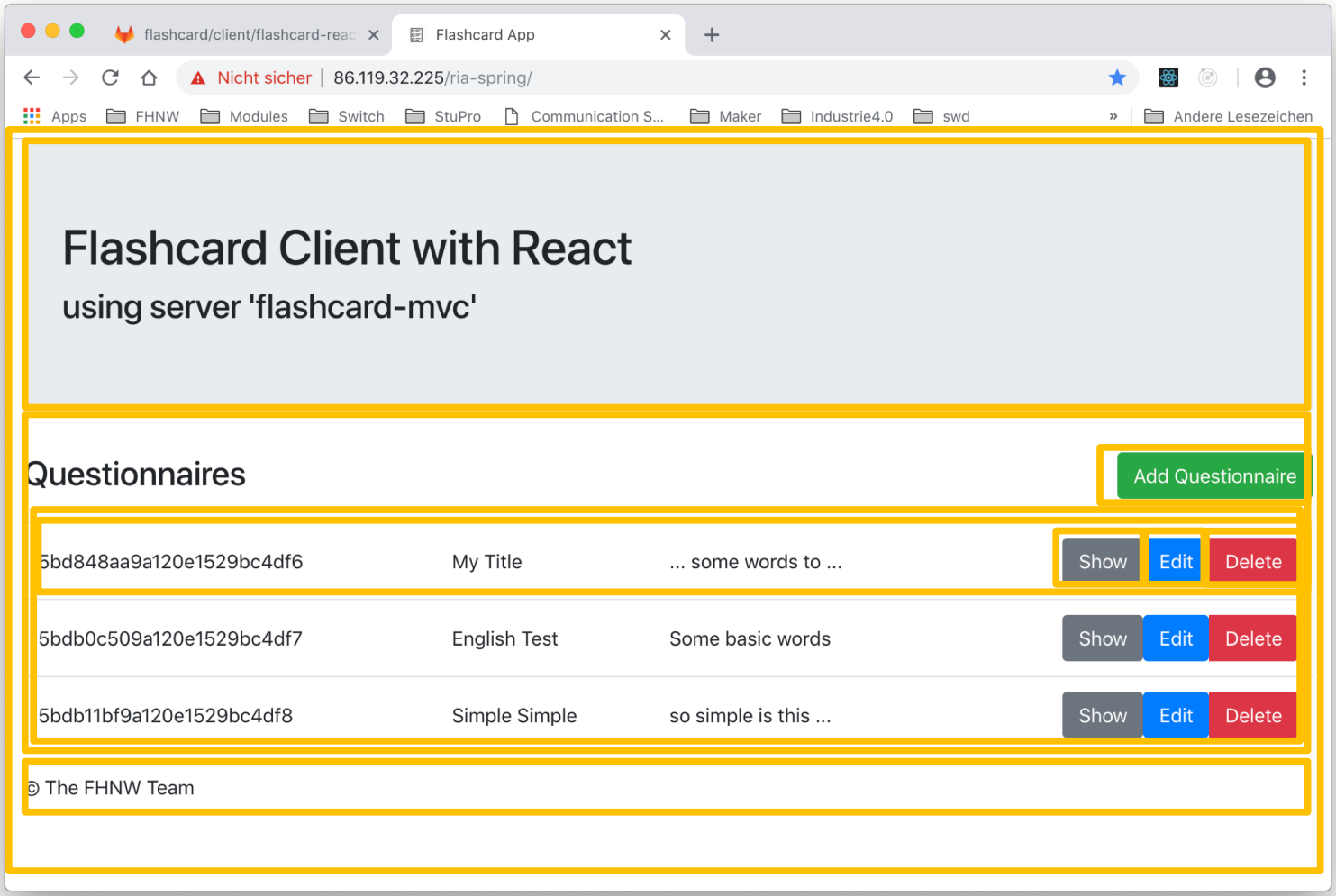


Typen von React Komponenten

	Presentational Components	Container Components
Zweck	**How things look** markup, styles	**How things work** Data Fetching, State Updates
Daten lesen	aus Properties	aus State/Properties
State	selten, wenn ausschliesslich für UI Komponenten	Applikation State
Type	Functional Component	Class Component

siehe https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

Dekomposition der Flashcard App (1/2)



Dekomposition der Flashcard App (1/2)

- Komponentenbaum (vereinfacht!)

  - **App**
    - **Header**
    - **QuestionnaireContainer**
      - **QuestionnaireCreateDialog**
      - **QuestionnaireTable**
        - **QuestionnaireTableElement**
          - **QuestionnaireShowDialog**
          - **QuestionnaireUpdatedialog**
    - **Footer**

Übung 7 als Hausaufgabe

- "Flashcard App" erweitern (**read-only!**), um
  - QuestionnaireContainer
  - QuestionnaireTable
  - QuestionnaireTableElement