React

Javascript library for building user interfaces

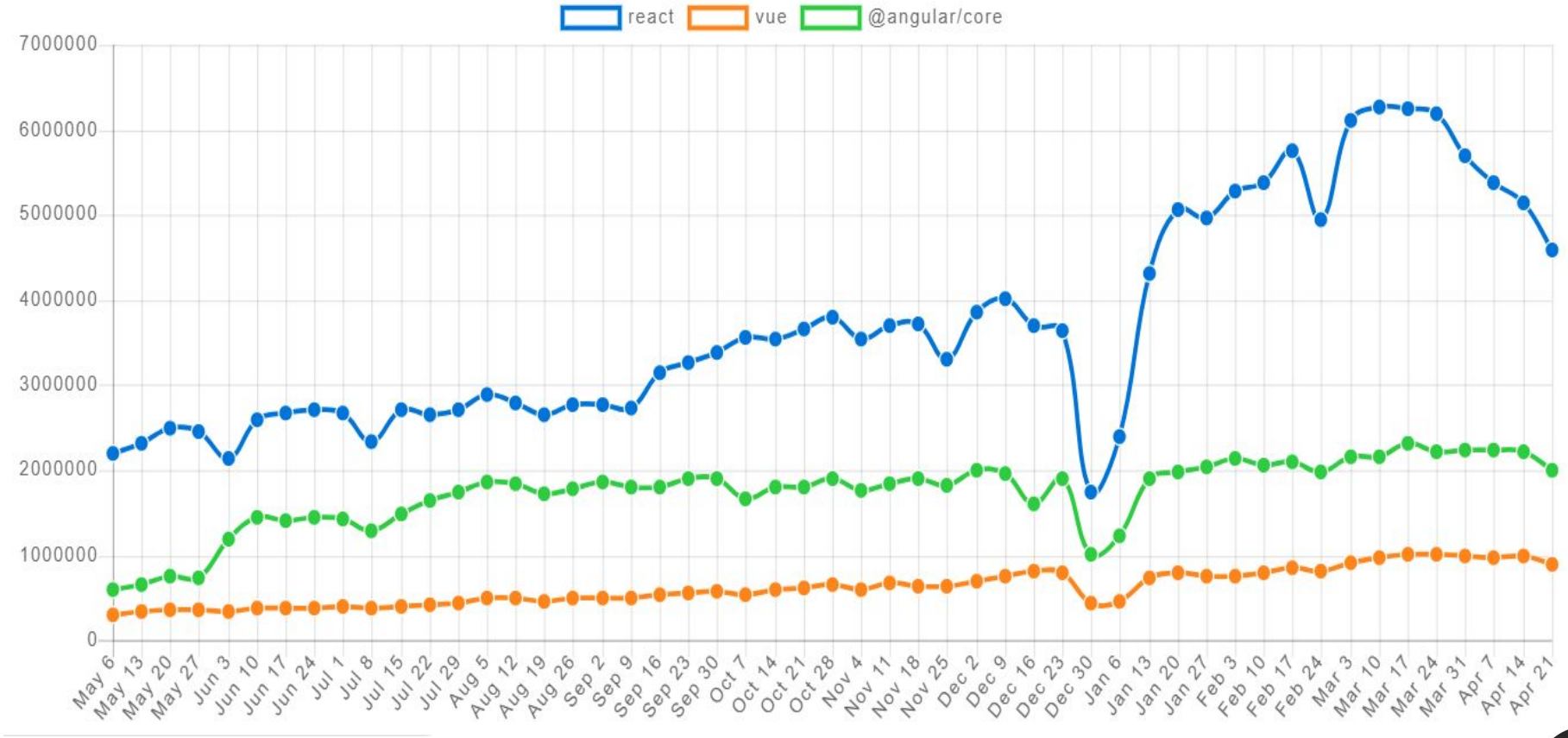
Čo je to React?

JS knižnica - ES6

Vytvorený Facebookom v roku 2012

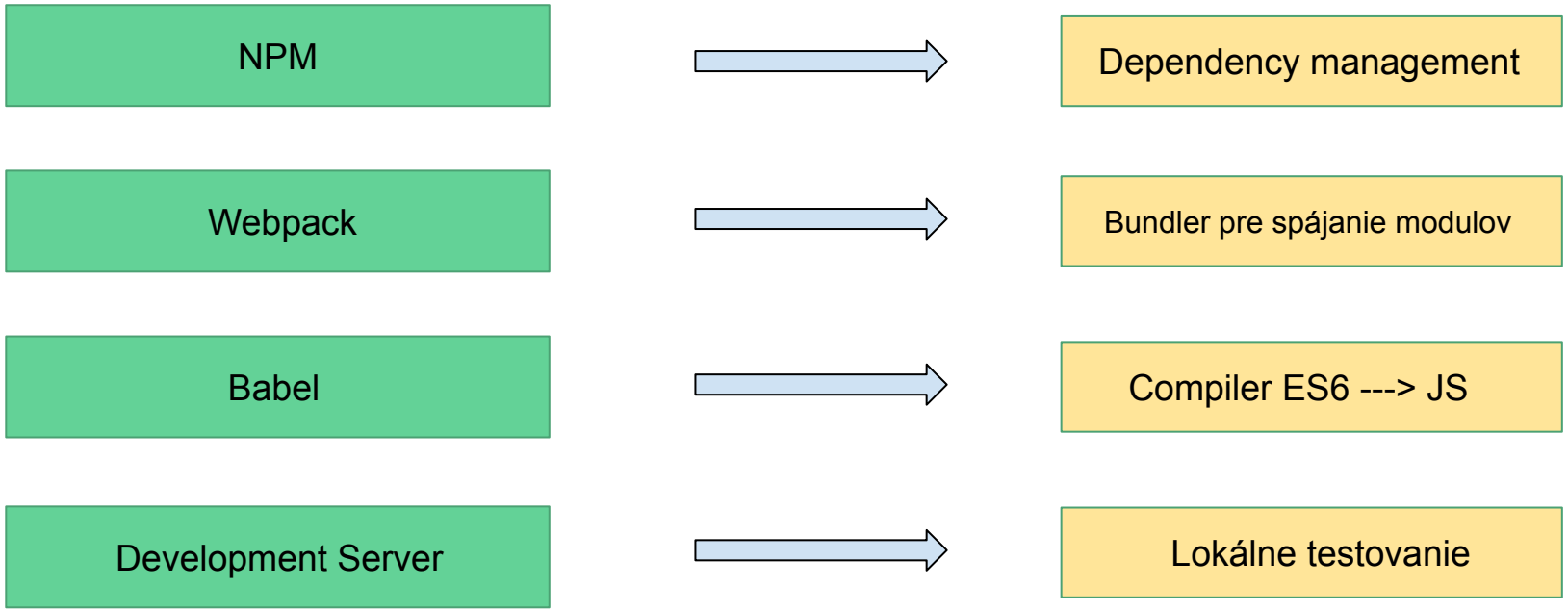
Top 3 js framework

Založený na komponentoch



Zdroj: npmtrends.com

Build Workflow



React-scripts

create-react-app - vytvorí nakonfigurovanú react aplikáciu

```
> npm install create-react-app -g

> create-react-app nazov-aplikacie
```

npm start - build a spustenie aplikácie

Čo dnes vytvoríme?

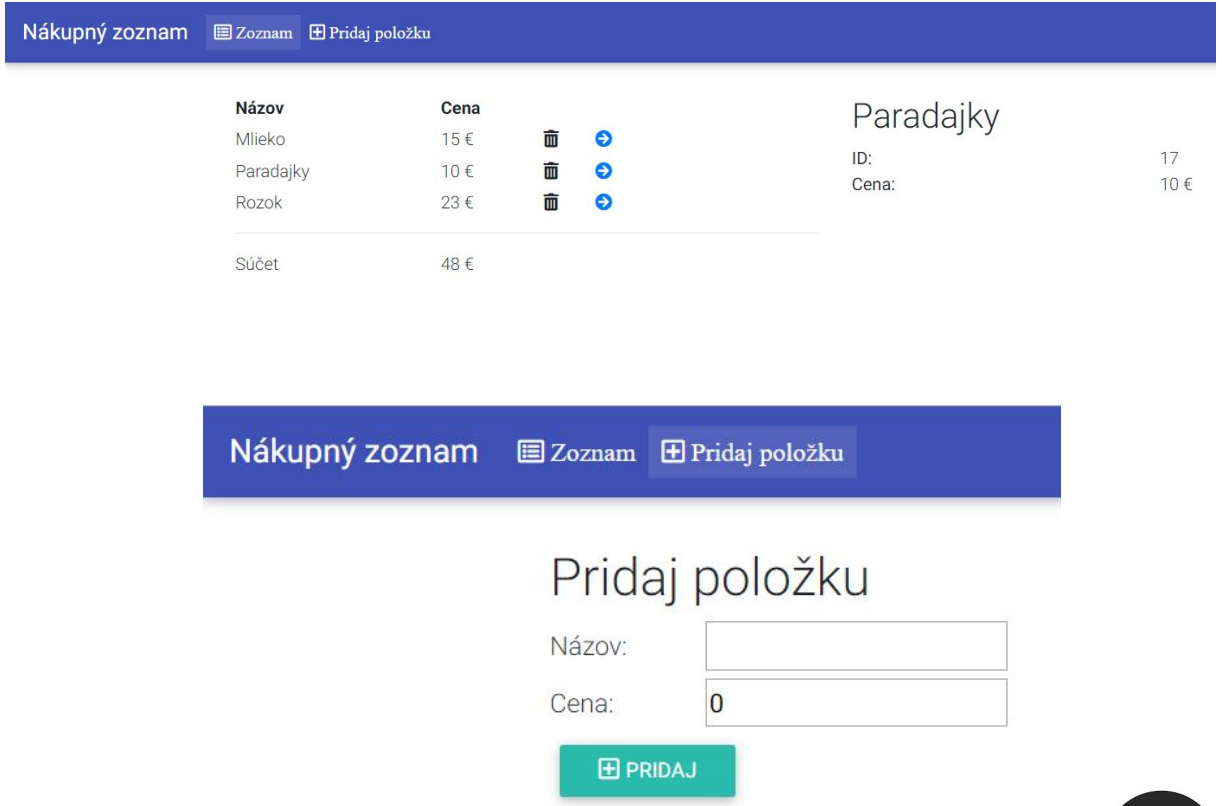
Nákupný zoznam

pridať tovar

prezerať si zoznam

odstrániť položku

zobraziť detail položky



Niečo málo o backende

NodeJs REST API - http://localhost:3000/api/

Databázová tabulka **ITEM**

4 metódy

POST items/ - Vytvorenie položky

GET items/ - Získanie všetkých položiek

GET items/:id - Získanie detailov o položke s id = :id

DELETE items/:id - Zmazenie položky s id = :id

Použité knižnice

ExpressJs - framework pre tvorbu web aplikácií

Sequelize - ORM pre nodeJS aplikácie

CORS - manažment Cross-Origin Resource Sharing

Helmet - zebezpečenie NodeJS web aplikácií

Winston - logovanie

Module alias - vykúpenie z ../../../../ hell

Komponenty

Custom “HTML” komponenty

Prepoužiteľné v aplikácií

ES6 trieda ktorá dedí od **React.Component**

Musí obsahovať **render()** funkciu

2 druhy komponentov

- **Funkčné komponenty** (Functional)
- **Komponenty založené na triedach** (Class-based)

Class-based komponenty

dedia od **React.Component**

dokážu reagovať na **funkcie životného cyklu** komponentu

udržujú svoj **stav**

obsahujú funkciu **render()**

```
import React from 'react'

class ItemDetail extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            item : null
        }
    }
     componentDidUpdate(){
        //Do something
     }

     render(){

         return(
             <div>
                 <p>Toto je Text</p>
             </div>
         );
     }
}

export default ItemDetail;
```

Functional components

Funkcia ktorá **vracia JSX** na zobrazenie

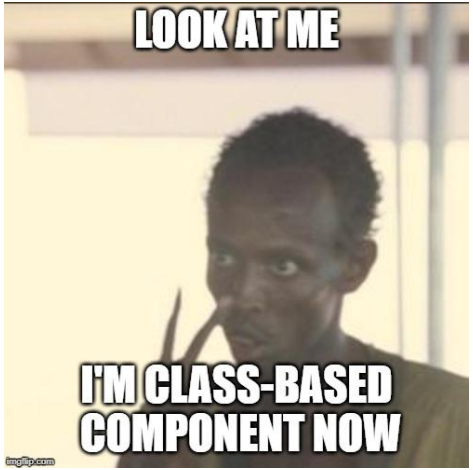
Nedokáže udržať stav

Nedokáže reagovať na lifecycle eventy

Použiť ak spomenuté nepotrebujeme

```jsx
import React from 'react'


const ListHeader = () => {
  return (
    <div className="row header">
        <div className="col-sm-4">Názov</div>
        <div className="col-sm-2">Cena</div>
        <div className="col-sm-3"></div>
        <div className="col-sm-3"></div>
    </div>
  )
}

export default ListHeader;
```

React-Hooks



Pre Class-based komponenty umožňujú

- udržiavať a meniť stav
- obsluhovať metódy životného cyklu
- useEffect, useState, useContext

JSX

syntaktické rozšírenie JavaScriptu

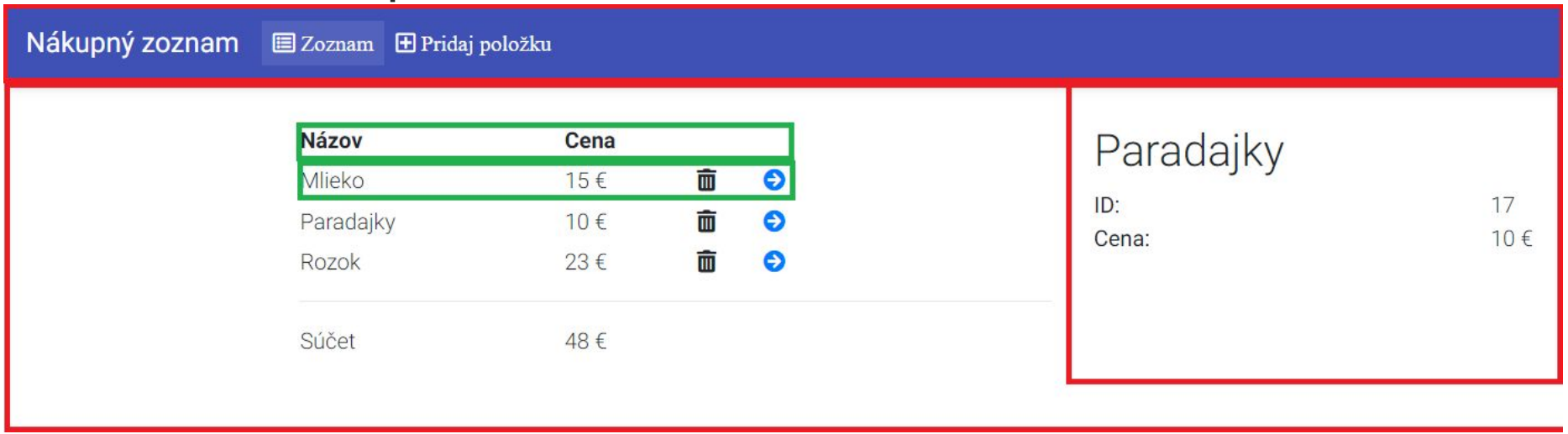
popisuje **vzhľad** komponentu

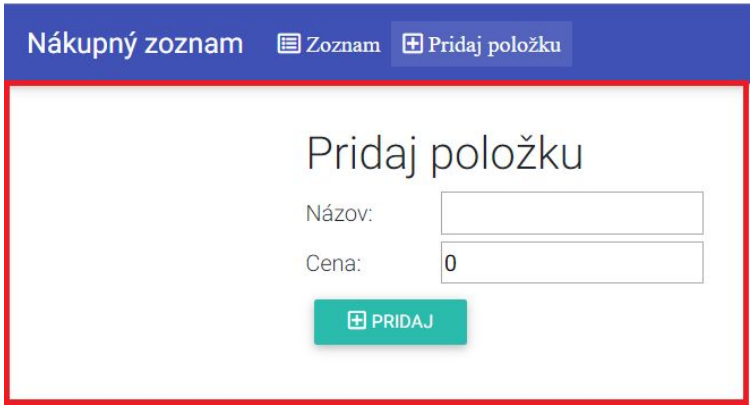
preložený do JavaScriptu

dynamické renderovanie komponentov

```jsx
import React from 'react'


const ListHeader = () => {
    let innerElement = (
        <p>Toto Je text</p>
    )

    const number = 4
    return (
        <div>
            {innerElement}
            <br/>
            {number}
        </div>
    )
}

export default ListHeader;
```

Detekcia komponentov





Class-based	Functional
Menu	ListItem
ShoppingList	ListHeader
AddItem	
ItemDetail	

Properties

vstupné parametre komponentu

zapisujeme ich ako **html atribúty**

môžu byť hodnotou aj funkciou

Získame z premennej **props** a pri class-based componentoch z **this.props**

```
import React from 'react';
import './ListItem.css';
import {NavLink}  from 'react-router-dom';


const ListItem = (props) => {

    return(

      <div className="row list-item">
          <div className="col-sm-4">{props.name}</div>
          <div className="col-sm-2">{props.price} &euro;</div>
          <div className="col-sm-1">
            <i className="fas fa-trash-alt"
               onClick={props.clickDelete}></i>
          </div>
          <div className="col-sm-1">
            <NavLink to={`/${props.id}`}
            className="fas fa-arrow-circle-right"></NavLink>
          </div>
      </div>
    )
}


export default ListItem;
```

State

obsahuje stav komponentu

nastavíme ho volaním **setState()**

čítame pomocou **this.state**

volanie metódy setState
zavolá **render()**

dáta z API, v akom stave je UI, ...

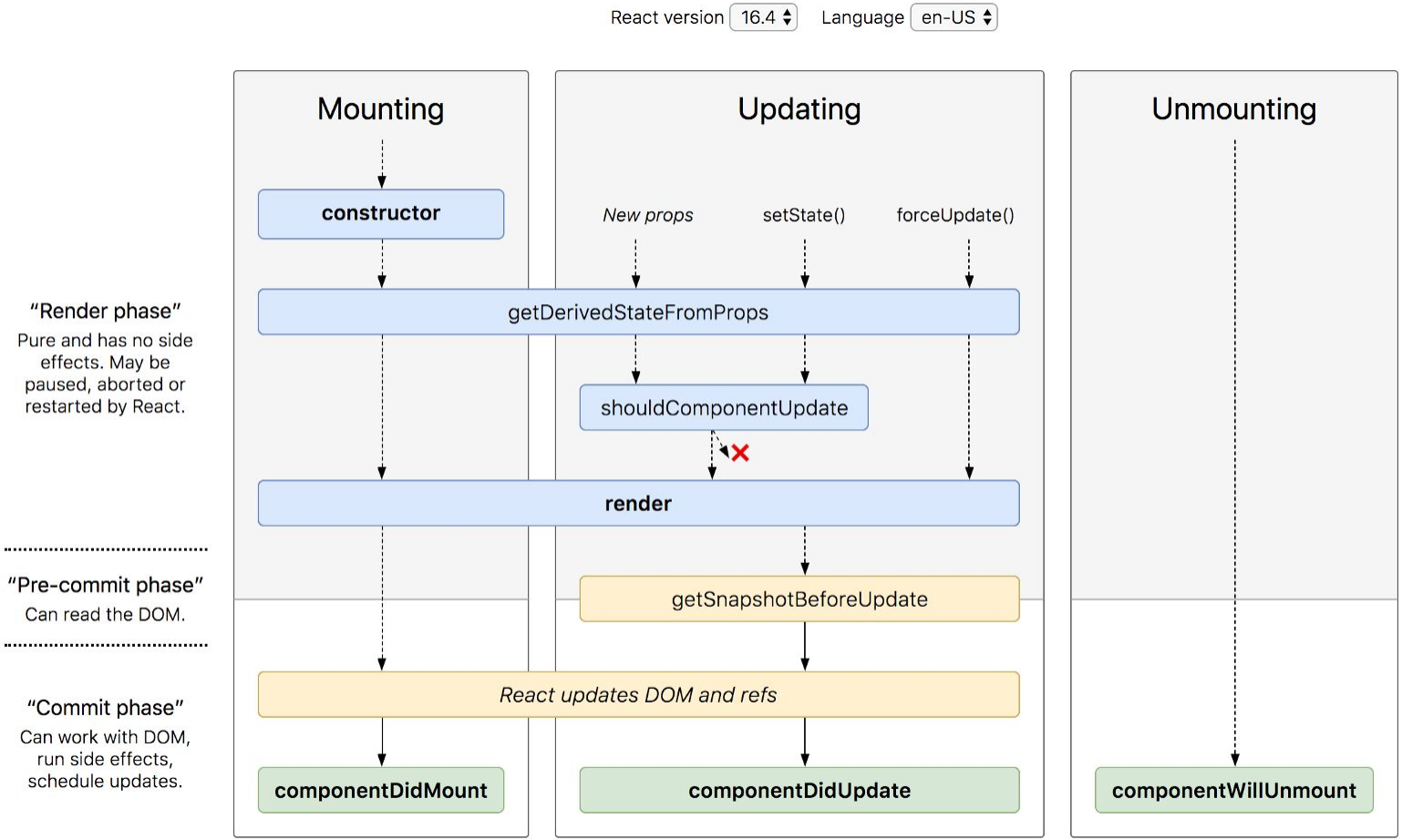
```
class ItemDetail extends React.Component{

    constructor(props){
        super(props);
        this.state = { item : null, text: 'AAA' }
    }
     componentDidMount(){
        this.setState({item:{
                id: 15
            }});
     }

     render(){
         let itemDetail = (<p>Načítavam</p>);
         if(this.state.item){
             itemDetail = (
                     <div>
                         <div>{this.state.text}</div>
                         <div>{this.state.item.id}</div>
                     </div>
             );
         }
         return(<div>
            {itemDetail}
            </div>);
     }
```

Stav v našej aplikácií

Potrebné správne určiť **ktorý komponen**t bude udržovať **ktorý stav**

Menu	activeMenuItem
ShoppingList	shoppingItems
AddItem	name, price
ItemDetail	item
ListItem	-
ListHeader	-

Životný cyklus komponentov



CSS a React

Každý komponent môže v sebe importovať css súbor

```
import React from 'react';
import './ShoppingList.css'
```

Platí pre **celú aplikáciu**

css triedy sa priraďujú pomocou **className**

Radium - Dynamické vytváranie štýlov a ukladanie do premenných

CssModules - prístup k css súborom ako k objektom

```
<MDBNavItem active= {this.state.active === 2} onClick = {this.handleMenuClick.bind(this,2)}>
    <MDBNavLink className="far fa-plus-square" to="/add"> Pridaj položku</MDBNavLink>
</MDBNavItem>
```

Routing

v SPA prebieha smerovanie na **Frontende**

react-router - obsahuje komponenty pre Routing

- Link
- Switch
- Route
- BrowserRouter

```
D:\Projekty\Prez>npm install --save react-router react-router-dom
```

BrowserRouter

HTML5 **history API**

pushState, popState, replaceState

Obaľuje časť aplikácie kde chceme využívať routing

Ao vnorené môžeme využívať:

- Link
- Switch
- Route

```
import ShoppingList from './shopping-list/ShoppingList'
import { BrowserRouter as Router, Switch, Route,} from 'react-router-dom';
import AddItem from './item-add/AddItem';


class App extends Component {
  render() {
    return (
        <Router>
            <div>
              <Menu />
              <div className="app container">
                <Switch>
                    <Route path='/add' component={AddItem} />
                    <Route path='/'  component={ShoppingList} />
                </Switch>
              </div>
            </div>
        </Router>

    );
  }
}

export default App;
```

Route

renderovanie komponentu ak je splnená **path podmienka**

atribút path definuje ako očakávame že bude **url začínať**

exact - url musí presne sedieť

Aktívnych môže byť **viac Route**

```
import ShoppingList from './shopping-list/ShoppingList'
import { BrowserRouter as Router, Switch, Route,} from 'react-router-dom';
import AddItem from './item-add/AddItem';


class App extends Component {
  render() {
    return (
        <Router>
            <div>
              <Menu />
              <div className="app container">
                <Switch>
                    <Route path='/add' component={AddItem} />
                    <Route path='/'  component={ShoppingList} />
                </Switch>
              </div>
            </div>
        </Router>

    );
  }
}

export default App;
```

Switch

zabezpečí, že aktívny bude len jeden detský Route

kontrola sa vykonáva v poradí v akom sú Route zapísané

prvý Route ktorý splní podmienku je vykreslený

```
import ShoppingList from './shopping-list/ShoppingList'
import { BrowserRouter as Router, Switch, Route,} from 'react-router-dom';
import AddItem from './item-add/AddItem';


class App extends Component {
  render() {
    return (
        <Router>
            <div>
              <Menu />
              <div className="app container">
                <Switch>
                    <Route path='/add' component={AddItem} />
                    <Route path='/'  component={ShoppingList} />
                </Switch>
              </div>
            </div>
        </Router>

    );
  }
}

export default App;
```

Link(NavLink)

Presmerovanie aplikácie

Adresa je zadaná v atribúte **to**

Každá adresa je absolútna

ak chceme využiť relatívnu adresu použijeme props

```
to={props.match.url +`${props.id}`}
```

```
render(){
    return (
            <MDBNavbar color="indigo" dark expand="md">
                <MDBNavbarBrand>
                    <strong className="white-text">Nákupný zoznam</strong>
                </MDBNavbarBrand>
                <MDBCollapse id="navbarCollapse3" isOpen={true} navbar>
                    <MDBNavbarNav left>
                        <MDBNavItem active= {this.state.active === 1}
                         onClick = {this.handleMenuClick.bind(this,1)}>
                            <MDBNavLink className="far fa-list-alt" to="/">
                             Zoznam
                             </MDBNavLink>
                        </MDBNavItem>
                        <MDBNavItem active= {this.state.active === 2}
                         onClick = {this.handleMenuClick.bind(this,2)}>
                            <MDBNavLink className="far fa-plus-square" to="/add">
                             Pridaj položku
                             </MDBNavLink>
                        </MDBNavItem>
                    </MDBNavbarNav>
                </MDBCollapse>
            </MDBNavbar>
    );
}
```

Posielanie parametrov

Route ktorý sa aktivuje pri url s path parametrom :id

```
<div className="col-sm-6">
    <Route path={this.props.match.url +  ':id'} exact component={ItemDetail} />
</div>
```

Ako napr `to={props.match.url +`${props.id}`}`

Získanie atribútu v komponente

```
const itemId = this.props.match.params.id;
```

Webová komunikácia

2 najznámejšie knižnice

- FETCH
  - jednoduché používanie
  - vstavaný v Reacte
- AXIOS
  - automatická konverzia na JSON
  - globálna konfigurácia, interceptory

Fetch

url ako vstupný parameter

vracia výsledok ako promise

musí byť transformovaný do JSON

Podporuje všetky HTTP metódy

```
componentDidMount(){
    this.getItemsFromServer();
}

getItemsFromServer(){
    fetch('http://localhost:3000/api/items')
        .then(result=>result.json())
        .then(resp=>this.setState({items: resp.data}));
}

handleClickDelete(id){

    fetch('http://localhost:3000/api/items/' + id, {
        method: 'DELETE',
    })
    .then(res => res.json())
    .then(res => {
        this.getItemsFromServer();
        console.log(res)}
    );


}
```

Axios

podporuje všetky http metódy

axios.[http-metoda]

ako parameter URL

výsledok vrátený ako JSON promise

```
import React from 'react';

import axios from 'axios';

export default class PersonList extends React.Component {
  state = {
    persons: []
  }

  componentDidMount() {
    axios.get(`https://jsonplaceholder.typicode.com/users`)
      .then(res => {
        const persons = res.data;
        this.setState({ persons });
      })
  }

  render() {
    return (
      <ul>
        { this.state.persons.map(person => <li>{person.name}</li>
      </ul>
    )
  }
}
```

Formuláre

V state uložíme dátovú štruktúru komponentu

Každú zmenu zapíšeme

Po stlačení submit odošleme pomocou fetch/axios http request s dátami

Validácia prebieha v metóde kde spracovávame zmenu

```
handleChange(event,key){
    this.setState({
        [key]: event.target.value
    });
}
render(){

    return(
      <div className="centered">
        <h2>Pridaj položku</h2>
        <form onSubmit={this.handleSubmit.bind(this)}>
          <label for="name"> Názov:</label>
          <input type="text" name="name"
                 onChange={(event) => this.handleChange(event,"name")}
                 value={this.state.name}/>
          <br/>
          <label for="price">Cena:</label>
          <input type="text" name="price"
                 onChange={(event) => this.handleChange(event,"price")}
                 value={this.state.price}/>
          <br/>
          <button type="submit"
           className="btn btn-default" id="add-button-form"
            value="Pridaj"><i className="far fa-plus-square mr-1">
            </i>Pridaj</button>
        </form>
      </div>
   );
```

React vs Angular

React

- Typescript
- Iba View - väčšia voľnosť
- JSX
- Create React App(react-scripts)
- aktuálne používanejší

Angular

- TypeScript
- Kompletná štruktúra MVC
- HTML Template
- Angular CLI
- Angular Forms
- Dependency Injection

Známe React weby

AirBnB https://cs.airbnb.com/

Atlassian https://www.atlassian.com/

Dropbox https://www.dropbox.com/

Facebook https://www.facebook.com/

Netflix https://www.netflix.com/browse

PayPal https://www.paypal.com/

Zdroje

Udemy Skolenie - React The Complete Guide
https://www.udemy.com/react-the-complete-guide-incl-redux/

React homepage
https://reactjs.org/

Ďakujem za pozornosť