

| React

TRENING DLA POCZĄTKUJĄCYCH

AGENDA

9:00



WPROWADZENIE

KOMPOZYCJA KOMPONENTÓW

PRZERWA

ROUTING

13:00



PRZERWA

KORZYSTANIE Z API

PRZERWA

STYLOWANIE I OBSŁUGA ZDARZEŃ

17:00



PRZYDATNE LINKI

NODE JS: <https://nodejs.org/en/>

DOKUMENTACJA REACT: <https://reactjs.org/docs/hello-world.html>

CREATE REACT APP: <https://github.com/facebook/create-react-app>

STYLED COMPONENTS: <https://www.styled-components.com/>

Ecma Script 6

WPROWADZENIE

Czym jest EcmaScript6?

Czym jest EcmaScript6?

Nowy standard języka **JAVASCRIPT**.

Nie jest natywnie wspierany przez wszystkie **przeglądarki**.

Pełna zgodność jest osiągane poprzez transpilację do starszej wersji (**EcmaScript5**).

Klasy w ES6

```
class Animal {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(this.name + ' makes a noise.');
```

Stałe i zmienne w ES6

```
let exampleVariable = 5;  
exampleVariable = 10;
```

```
const exampleConstants = 5
```

```
const examplePearson = {  
  name: 'Jan',  
  surname: 'Kowalski',  
};
```

```
examplePearson.surname = 'Lewandowski';
```


Arrow function w ES6

```
const up1 = (param) => {  
  return param.toUpperCase()  
};  
  
const up2 = (param) => param.toUpperCase();  
  
const up3 = param => param.toUpperCase();  
  
console.log(up1('abcd'))  
  
// ABCD
```

Template Strings w ES6

```
const name = 'Chris';

const message = `Hello ${name}`;

console.log(message);

// Hello 'Chris'
```

```
const multilineMessage =
`first line
second line
third line`;

console.log(multilineMessage);

// first line
// second line
// third line
```



Create React App

BOILERPLATE

Node + NPM + Yarn

NodeJS: <https://nodejs.org>

Yarn: <https://yarnpkg.com>

Create React App - boilerplate

Create React App: <https://github.com/facebook/create-react-app>

Zalecana instalacja z wykorzystaniem **YARN**.

ISSUE #0

Zainstaluj środowisko NODE i YARN.

Uruchom nowy projekt **React** korzystając z **create-react-app**.

ISSUE #1

Zmodyfikuj projekt, tak żeby wyświetlał tylko nazwę Twojego bloga.

Usuń niepotrzebne pliki.

Zaleca się korzystanie z kontroli wersji.

Kompozycja komponentów

MODUŁ 1



SINGLE PAGE APPLICATION

SINGLE PAGE APPLICATION

Podejście, w którym jednokrotnie pobieramy całą aplikację, obsługa podstron i innych elementów jest obsługiwana poprzez wstrzykiwanie elementów drzewa **DOM** przez **JAVASCRIPT**.

SINGLE PAGE APPLICATION

Podejście, w którym jednokrotnie pobieramy całą aplikację, obsługa podstron i innych elementów jest obsługiwana poprzez wstrzykiwanie elementów drzewa **DOM** przez **JAVASCRIPT**.

Strony i aplikacje wykonane w technologii **SPA** działają znacząco **SZYBCIEJ**.

SINGLE PAGE APPLICATION

Podejście, w którym jednokrotnie pobieramy całą aplikację, obsługa podstron i innych elementów jest obsługiwana poprzez wstrzykiwanie elementów drzewa **DOM** przez **JAVASCRIPT**.

Strony i aplikacje wykonane w technologii **SPA** działają znacząco **SZYBCIEJ**.

Znacząco mniej obciążają **SERWERY**.

Minusy podejścia SINGLE PAGE APPLICATION

Minusy podejścia SINGLE PAGE APPLICATION

Część wyszukiwarek gorzej indeksuje strony - problem [SEO](#).

Minusy podejścia SINGLE PAGE APPLICATION

Część wyszukiwarek gorzej indeksuje strony - problem [SEO](#).

Problemy w [SOCIAL MEDIA](#). Facebook nie zaczytuje tagów podstron.

Czym jest REACT JS

React JS

Biblioteka do tworzenia interfejsów użytkownik

React Router

Redux

Biblioteka do zarządzania stanem aplikacji

MobX

Biblioteka do zarządzania stanem aplikacji

Redux Saga

Immutable JS

Redux Thunk

Czym jest JSX?

Czym jest JSX?

Syntax pozwalający na łatwe używanie elementów **HTML** wewnątrz **JS**.

Jest transpilowany do **JAVASCRIPTU** - biblioteka **BABEL**.

Czym jest JSX?

Kod **JS**:

```
const App = (value) => (  
  React.createElement(  
    'div',  
    {},  
    React.createElement(  
      'h1',  
      {},  
      value  
    )  
  )  
);
```

Kod **JSX**:

```
const App = (value) => (  
  <div>  
    <h1>{value}</h1>  
  </div>  
);
```

Czym jest JSX?

Używając **JSX**, należy zwrócić uwagę na nazwy niektórych **properties'ów**.

```
const App = (value) => (  
  <div className="container">  
    <h1 className="container__header">{value}</h1>  
  </div>  
);
```

Komponenty

Podstawowy budulec aplikacji **React**.

Aplikacja w **React** jest tworzona za pomocą re-używalnych **komponentów**.

Komponenty

Komponenty **React** mogą być zbudowane przy użyciu:

FUNKCJI

```
import React from 'react';

function Header() {
  return <div>Lorem Ipsum</div>
}

export default Header;
```

```
import React from 'react';

const Header = () => (<div>Lorem Ipsum</div>);

export default Header;
```

KLASY ES6 rozszerzonej o React.Component

```
import React, { Component } from 'react';

class Header extends Component {
  render() {
    return (
      <div>Lorem Ipsum</div>
    )
  }
}

export default Header;
```

Komponenty - użycie

Komponent Header

```
import React from 'react';

const Header = () => (
  <div>
    Lorem Ipsum
  </div>
);

export default Header;
```

Główny komponent App

```
import React, { Component } from 'react';

import Header from './header.component.js';

class App extends Component {
  render() {
    return (
      <div>
        <Header />

        <div>Site content - lorem ipsum</div>
      </div>
    )
  }
}

export default App;
```

Komponenty - obiekt props

Jest to **niemutowalny** obiekt komponentu.

Wartości obiektu **props** są przekazywane do niego jako **atrybuty**.

Przekazywane mogą być zarówno **wartości**, **obiekty** jak i referencje **funkcji**.

Komponenty - obiekt props

Komponent Header

```
import React from 'react';

const Header = (props) => (
  <div>
    <h1>{props.title}</h1>
    <h2>{props.subtitle}</h2>
  </div>
);

export default Header;
```

Główny komponent App

```
import React, { Component } from 'react';

import Header from './header.component.js';

class App extends Component {
  render() {
    return (
      <div>
        <Header
          title="Lorem"
          subtitle="Ipsum"
        />

        <div>Site content - lorem ipsum</div>
      </div>
    )
  }
}

export default App;
```

Komponenty - obiekt props

Komponent Header

```
import React, { Component } from 'react';

class Header extends Component {
  render() {
    return (
      <div>
        <h1>{this.props.title}</h1>
        <h2>{this.props.subtitle}</h2>
      </div>
    )
  }
}

export default Header;
```

Główny komponent App

```
import React, { Component } from 'react';

import Header from './header.component.js';

class App extends Component {
  render() {
    return (
      <div>
        <Header
          title="Lorem"
          subtitle="Ipsum"
        />

        <div>Site content - lorem ipsum</div>
      </div>
    )
  }
}

export default App;
```

Komponenty - props children

Komponent Header

```
import React, { Component } from 'react';

class Header extends Component {
  render() {
    return (
      <div>
        {this.props.children}
      </div>
    )
  }
}

export default Header;
```

Główny komponent App

```
import React, { Component } from 'react';

import Header from './header.component.js';

class App extends Component {
  render() {
    return (
      <div>
        <Header>
          <h1>Lorem Ipsum</h1>
        </Header>

        <div>Site content - lorem ipsum</div>
      </div>
    )
  }
}

export default App;
```

ISSUE #2

Wykorzystując kod z poprzedniego zadania, odtwórz wygląd **designu**.

Postaraj się utworzyć komponenty HEADER, FOTTER oraz TEXT-SECTION, który jest komponentem re-używalnym. Dane komponentu powinny być dostarczane z wykorzystaniem obiektu props.

Mój Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse fringilla fringilla neque ac laoreet. Fusce vel nulla ornare, aliquam ex eget, sollicitudin neque. Morbi venenatis rutrum ligula vel scelerisque.

Lorem ipsum dolor.

© Copyright - 2018.

Komponenty - funkcje komponentów

ES 6

```
import React, { Component } from 'react';

class App extends Component {
  constructor( props ){
    super( props );
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() { console.log('click button')};

  render() {
    return (
      <div>
        <button onClick={this.handleClick} >
          Click
        </button>
      </div>
    )
  }
}
```

ES 6+

```
import React, { Component } from 'react';

class App extends Component {

  handleClick = () => { console.log('click button')};

  render() {
    return (
      <div>
        <button onClick={this.handleClick} >
          Click
        </button>
      </div>
    )
  }
}
```

Komponenty - stan komponentu

Komponenty - stan komponentu

Obiekt, który może być **mutowany**, przy użyciu metody **setState**.

Zmiana **stanu komponentu**, powoduje jego **odświeżenie** (re-render).

Stan komponentu jest dostępny tylko w **statefull components** (komponenty klasowe).

Komponenty - modyfikacja stanu komponentu

ES 6+

```
import React, { Component } from 'react';

class App extends Component {

  state = { count: 0, };

  incrementCounter = () => this.setState({count: this.state.counter + 1});

  render() {
    const { count } = this.state;

    return (
      <div>
        <div>Count: {count}</div>
        <button onClick={this.incrementCounter}>+</button>
      </div>
    )
  }
}
```


Komponenty - modyfikacja stanu komponentu

Funkcja `setState()` jest asynchroniczna!

Jeżeli modyfikujemy stan komponentu, wykorzystując, bieżącą wartość stanu, nie powinniśmy odczytywać go w locie.

```
incrementCounter = () => (  
  this.setState(prevState => ({  
    count: !prevState.count  
  }))  
);
```

ISSUE #3

Zaimplementuj obsługę licznika, wykorzystując re-używalne komponenty.

Zadbaj, żeby rok w stopce wyświetlał zawsze aktualną datę.

Mój Blog

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse fringilla fringilla neque ac laoreet. Fusce vel nulla ornare, aliquam ex eget, sollicitudin neque. Morbi venenatis rutrum ligula vel scelerisque.

Stan: 21

+

-

RESET

© Copyright - 2018.

Routing

MODUŁ 2

React Router

Nazwa biblioteki: `react-router-dom`

Biblioteka jest dostępna w NPM, wymagana będzie instalacja

```
$ npm install react-router-dom --save
```

lub

```
$ yarn add react-router-dom
```

React Router - utworzenie podstron

Import biblioteki:

```
import {BrowserRouter, Route} from 'react-router-dom';
```

Utworzenie podstron:

```
<BrowserRouter>
  <Switch>
    <Route exact path="/" component={Home} />
    <Route path="/about" component={AboutMe} />
    <Route path="/contact" component={Contact} />
    <Route path="/projects" component={Projects} />
  </Switch>
</BrowserRouter>
```

React Router - utworzenie podstron

Przykładowe osadzenie w pliku: `index.js`

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter, Route } from 'react-router-dom';

import Home from './Home';
import About from './About';

ReactDOM.render(
  <BrowserRouter>
    <div>
      <Route exact path="/" component={Home} />
      <Route path="/about" component={About} />
    </div>
  </BrowserRouter>,
  document.getElementById('root')
);
```

React Router - przekazywanie danych

Route z dynamicznym przekazaniem wartości: `id`

```
<BrowserRouter>
  <Switch>
    <Route exact path="/" component={Home} />
    <Route path="/about" component={About} />
    <Route path="/post/:id" component={Post}/>
  </Switch>
</BrowserRouter>,
```

Komponent `<Link />` przekazujący wartość: `id` .

```
<Link to={`/${post.id}`}>
  {post.title}
</Link>
```

React Router - przekazywanie danych

Przekazany parametr, będzie dostępny w obiekcie `props`

```
{this.props.match.params.id}
```


React Router - domyślny route

Route z dynamicznym przekazaniem wartości: `id`

```
import { BrowserRouter, Switch, Route } from 'react-router-dom';

<BrowserRouter>
  <Switch>
    <Route exact path="/" component={Home} />
    <Route component={Default} />
  </Switch>
</BrowserRouter>
```

React Router - przekierowanie route'a

Konkretny `<Route />` można przekierować używając:

```
<Redirect from='*' to='/' />
```

ISSUE #4

Utwórz podstrony: **Autor**, **Kontakt**, oraz dynamiczną podstronę, będącą widokiem postu.
Podstrony mają zawierać ten sam nagłówek oraz stopkę, ale różnić się treścią.

Utwórz komponent `<Menu />`, służący do zmiany podstron.

Na podstronie postu - wyświetl dynamicznie przekazane ID.

EXTRA ISSUE #1

Utwórz podstronę 404, informującą o niedostępności adresu URL.

Po 10s, przekieruj użytkownika, ze strony 404 na stronę startową.

Korzystanie z API

MODUŁ 3

Cykl życia komponentów

Montowanie

```
componentDidMount()
```

```
render()
```

```
UNSAFE_componentWillMount()
```

Odświeżanie

```
componentDidUpdate()
```

```
render()
```

```
UNSAFE_componentWillUpdate()
```

Odmontowanie

```
componentWillUnmount()
```

Pobieranie danych z API

Pobranie danych z API, zazwyczaj wykonuje podczas wywoływania się metody:

`componentDidUpdate()`

```
componentDidMount() {  
  fetch('https://jsonplaceholder.typicode.com/todos/1')  
    .then(response => response.json())  
    .then(data => this.setState({  
      postsList: data,  
    }));  
};
```

Renderowanie obiektów tablicy

```
class App extends Component {
  state = {
    arrayList: ['aaa', 'bbb', 'ccc', 'ddd'],
  };

  renderList = () => this.state.arrayList.map((item, index) => <div key={index}>{item}</div>);

  render() {
    return (
      <div>
        {this.renderList()}
      </div>
    );
  }
}
```

Atrybut **key**, musi być wartością unikalną. Nie zaleca się stosować wartości **index**.

ISSUE #5

Na podstronie głównej, utwórz komponent wyświetlający listę postów oraz wykonaj zapytanie do API, w celu pobrania listy postów.

Na wcześniej utworzonym widoku postu, należy wyświetlić jego szczegóły.

EXTRA ISSUE #2

Przy użyciu stanu aplikacji, podczas ładowania danych wyświetl loader lub napis "ładowanie". Dla łatwości implementacji, możesz dodać 10s opóźnienia.

Praca z formularzami

```
class ExampleForm extends Component {
  state = { value: '', }

  handleChange = (event) => {
    this.setState({value: event.target.value});
  }

  handleSubmit = (event) => {
    console.log('Form submitted: ' + this.state.value);
    event.preventDefault();
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

ISSUE #6

Utwórz na podstronie posta formularz służący do dodawania komentarzy.

Po dodaniu komentarza, powinien wyświetlić on się pod postem.

EXTRA ISSUE #3

Komentarze każdego posta, powinny zostać zachowane w **localStorage**, tak żeby po odświeżeniu strony były one ponownie widoczne.

Stylowanie i obsługa zdarzeń

MODUŁ 4

Inline styles

Podstawowym zaimplementowanym mechanizmem do obsługi wyglądu, są **inline-styles**.

```
import React, { Component } from 'react';

const divStyle = {
  fontSize: '15px',
  textAlign: 'center'
};

class Header extends Component {
  render() {
    return (
      <div style={divStyle}>
        Lorem Ipsum
      </div>
    );
  }
}

export default Header;
```

CSS modules

Każdy komponent React otrzymuje własny **plik CSS**, który jest ograniczony do tego pliku i komponentu.

W celu wsparcia obsługi **css modules**, jest wymagana modyfikacja **create-react-app**.

```
import React, { Component } from 'react';
import styles from './App.css';

class App extends Component {
  render() {
    return (
      <div className={styles.App}>
        <div className={styles.header}>
          <h2>Welcome to React</h2>
        </div>
      </div>
    );
  }
}
```


Styled Components

Jest to biblioteka pozwalająca pisać kod CSS w ramach kodu JS. Tworzone są gotowe ostyleowane elementy.

Nazwa biblioteki: `styled-components`

Biblioteka jest dostępna w NPM, wymagana będzie instalacja

```
$ npm install styled-components --save
```

lub

```
$ yarn add styled-components
```

Styled Components

```
import React from 'react';
import styled from 'styled-components';

const Div = styled.div`
  margin: 40px;
  &:hover {
    background-color: yellow;
  }
`;

const Paragraph = styled.p`
  font-size: 15px;
  background-color: ${props => props.yellow ? 'yellow' : 'green'};
`;

const Container = () => (
  <Div>
    <Paragraph yellow={true}>Styled Paragraph</Paragraph>
  </Div>
);

export default Container;
```

Obsługa zdarzeń

Z poziomu react, możemy zapinać **eventy** na dowolnym elemencie drzewa dom.

```
import React, {Component} from 'react';

class App extends Component {

  componentDidMount() {
    window.addEventListener('scroll', this.handleScroll);
  }

  componentWillUnmount() {
    window.removeEventListener('scroll', this.handleScroll);
  }

  handleScroll = () => console.log(window.scrollY);

  render() {
    return (
      <div>
        Lorem Ipsum
      </div>
    );
  }
}
```

ISSUE #7

Wykorzystując bibliotekę: **styled-components**, utwórz style komponentów i wydziel je do zewnętrznych plików.

Nagłówek, podczas scrollowania powinien zmienić swoje style (position: fixed, top left: 0).