ReactJS

KOMPOZYCJA KOMPONENTÓW

Hello, World!

KAJ BIAŁAS
JavaScript Developer

Issue #0

Zaimplementuj aplikację zgodnie ze strukturą, korzystając z obiektu props komponentów:

Utwórz w aplikacji podstrony: `Home` `About` `Contact`

Issue #0

Sekcja z treścią powinna mieć rozmiar: `2000px`

Zaimplementuj mechanizm dock'owania menu podczas scroll'owania.

Komponenty

Pure Components

PureComponent jest podobny w działaniu do znanego wcześniej Component.

Różnice:

- nie mutuje danych komponentu
- podnosi perfomance aplikacji

Komponenty

Pure Components - użycie

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

class Header extends PureComponent {

    render() {
        return (
            <header>
                Header
            </header>
        );
    }
}

export default Header;
```

Komponenty

Pure Components - na co uważać?

Wprowadzanie danych w metodzie `render()`

```
render() {
  const { posts } = this.props
  const topTen = posts.sort((a, b) => b.likes - a.likes).slice(0, 9)
  return //...
}
```

Komponenty

Pure Components - dodatkowe informacje

https://codeburst.io/when-to-use-component-or-purecomponent-a60cfad01a81

Issue #1

Wykorzystując aplikację z: `Issue #0` , przerób dotychczasowe komponenty klasowe w użyciem: `PureComponents`

Komponenty

`<Fragment />`

Komponent `<Fragment />` jest rodzajem wrapper'a, przydatnym jeśli w funkcji `render()` chcemy mieć kontener, który nie przyjmuje, żadnych styli.

Jest wspierany od wersji 16.2

Komponenty

`<Fragment />` - korzyści?

Komponent `<Fragment />` sprawia, że w wy-renderowanym drzewie DOM, nie ma żadnego elementu

Komponenty

`<Fragment />` - użycie

```
import React, { Fragment } from 'react'

const App = () => (
  <Fragment>
    <p>foo</p>
    <p>bar</p>
  </Fragment>
)
```

```
import React, { Fragment } from 'react'

class App extends Component {
  render() {
    return(
        <Fragment>
            <p>foo</p>
            <p>bar</p>
        </Fragment>
    )
  }
}
```

Komponenty

`<Fragment />` - użycie

```
import React, { Fragment } from 'react'

const App = () => (
  <>
    <p>foo</p>
    <p>bar</p>
  </>
)
```

```
import React, { Fragment } from 'react'

class App extends Component {
  render() {
    return(
        <>
            <p>foo</p>
            <p>bar</p>
        </>
    )
  }
}
```

Komponenty

`<Fragment />` - alternatywa

```
class App extends Component {
  render() {
    return [<p>foo</p>, <p>bar</p>]
  }
}
```

metoda niezalecana

Issue #2

Wykorzystując aplikację z: `Issue #1` , wykorzystaj objekt:

`<Fragment />` do wyrenderowania wrapperów komponentów.

Komponenty

walidacja properties'ów

propTypes - typy właściwości - mechanizm

pozwalający sprawdzać poprawność props'ów

w komponentach

https://reactjs.org/docs/typechecking-with-proptypes.html

Komponenty

walidacja properties'ów

Nazwa biblioteki: `prop-types`

Instalacja biblioteki: `npm install prop-types --save`

Komponenty

`prop-types` użycie

```
import React from 'react';
import PropTypes from 'prop-types';

const Header = (props) => (
    <header>{props.applicationName}</header>
);

Header.propTypes = {
    aplicationName: PropTypes.string,
};

export default Header;
```

```
import React, { PureComponent } from 'react';
import PropTypes from 'prop-types';

class Header extends PureComponent {
    static propTypes = {
        applicationName: PropTypes.string,
    };

    render() {
        return (
            <header>
                {this.props.applicationName}
                </header>
        );
    }
}

export default Header;
```

Komponenty

`prop-types` - możliwe typy

Podstawowe:

`string` `number` `bool` `func`

Inne:

`array` `object` `element` `any`

Issue #3

Wykorzystując aplikację z: `Issue #2` , dodaj obsługę propTypes w istniejącym projekcie, dla każdego komponentu, przyjmującego props'y.

Sprawdź, co się stanie w przypadku, zadeklarowania niepoprawnego typu.

Komponenty

`prop-types` - isRequired

isRequired - rozszerzenie, sprawiające że dany properties jest wymagany

Komponenty

`prop-types` - isRequired (użycie)

```
import React, { PureComponent } from 'react';
import PropTypes from 'prop-types';

class Header extends PureComponent {
    static propTypes = {
        applicationName: PropTypes.string.isRequired,
    };

    render() {
        return (
            <header>
                {this.props.applicationName}
            </header>
        );
    }
}

export default Header;
```

Issue #4

Wykorzystując aplikację z: `Issue #3` , dodaj obsługę isRequired w istniejącym projekcie, dla każdego komponentu, przyjmującego props'y.

Sprawdź, co się stanie w przypadku, zadeklarowania niedostarczonego properties'u.

Komponenty

domyślne wartości props'ów

defaultProps - obiekt zawierający domyślne

wartości dla propertiesów komponentu

https://reactjs.org/docs/typechecking-with-proptypes.html

Komponenty

`defaultProps` użycie

```
import React, { PureComponent } from 'react';
import PropTypes from 'prop-types';

class Header extends PureComponent {
    static propTypes = {
        applicationName: PropTypes.string.isRequired,
    };

    static defaultProps = {
        applicationName: "example application name"
    };

    render() {
        return (
            <header>
                {this.props.applicationName}
                </header>
        );
    }
}

export default Header;
```

Komponenty

`defaultProps` użycie

```
import React from 'react';
import PropTypes from 'prop-types';

const Header = (props) => (
    <header>{props.applicationName}</header>
);

Header.propTypes = {
    aplicationName: PropTypes.string,
};

Header.defaultProps = {
    aplicationName: 'example name',
};

export default Header;
```

Issue #5

Wykorzystując aplikację z: `Issue #4` , dodaj obsługę defaultProps w istniejącym projekcie, oraz dodaj domyślne wartości.

Dodaj nowe properties dla nagłówka i stopki zawierające wersję aplikacji w nagłówku, oraz tekst odnośnie zastrzeżenia praw autorskich w stopce. Niech będa to wartości pobrane z default props.

Komponenty

```
this.props.children
```

Properties komponentu pozwalający wyrenderować dziecko komponentu.

Komponenty

`this.props.children` - użycie

```
import React, { PureComponent } from 'react';
import PropTypes from 'prop-types';

class Header extends PureComponent {
    render() {
        return (
            <header>
                {this.props.children}
            </header>
        );
    }
}

export default Header;
```

```
import React, { PureComponent } from 'react';
import Header from './Header';
import './App.css';

class App extends PureComponent {
  render() {
    return (
      <div className="App">
        <Header>
          Example title
        </Header>
      </div>
    );
  }
}

export default App;
```

Issue #6

Wykorzystując aplikację z: `Issue #5` , zmień komponent section tak aby korzystał z obiektu `this.props.children`

Komponenty

stan komponentu - użycie

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

class App extends PureComponent {
  state = {
        count: 0,
  };

  render() {
    return (
      <div className="App">
          Lorem ipsum
      </div>
    );
  }
}

export default App;
```

Komponenty

stan komponentu - użycie

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

class App extends PureComponent {
  constructor(props) {
    super(props);
    this.state = {count: 0};
  }

  render() {
    return (
      <div className="App">
          Lorem ipsum
      </div>
    );
  }
}

export default App;
```

Issue #8

Wykorzystując aplikację z: `Issue #7` , dodaj dodatkową podstronę: `Count`

Zaimplementuj funkcjonalność licznika kliknięć. Do deklaracji stanu wykorzystaj podejście z konstruktorem.

Komponenty

metody komponentu - użycie

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

class App extends PureComponent {
  state = {
        count: 0,
  };

  increaseCounter = () => {
    this.setState({
        count: this.state.count + 1,
    });
  }

  render() {
    return (
      <div className="App">
          <button onClick={this.increaseCounter}>Increase</button>
      </div>
    );
  }
}

export default App;
```

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

class App extends PureComponent {
  constructor(props) {
    super(props);
    state = {
        count: 0,
    };
    this.increaseCounter.bind(this)
  }


  increaseCounter() {
    this.setState({
        count: this.state.count + 1,
    });
  }

  render() {
    return (
      <div className="App">
          <button onClick={this.increaseCounter}>Increase</button>
      </div>
    );
  }
}

export default App;
```

Issue #9

Wykorzystując aplikację z: `Issue #8` , dodaj dodatkową funkcjonalność: `decreaseCounter` , służącą do zmniejszania stanu licznika.

Zaimplementuj funkcjonalność, wykorzystując podejście z klasycznymi metodami.