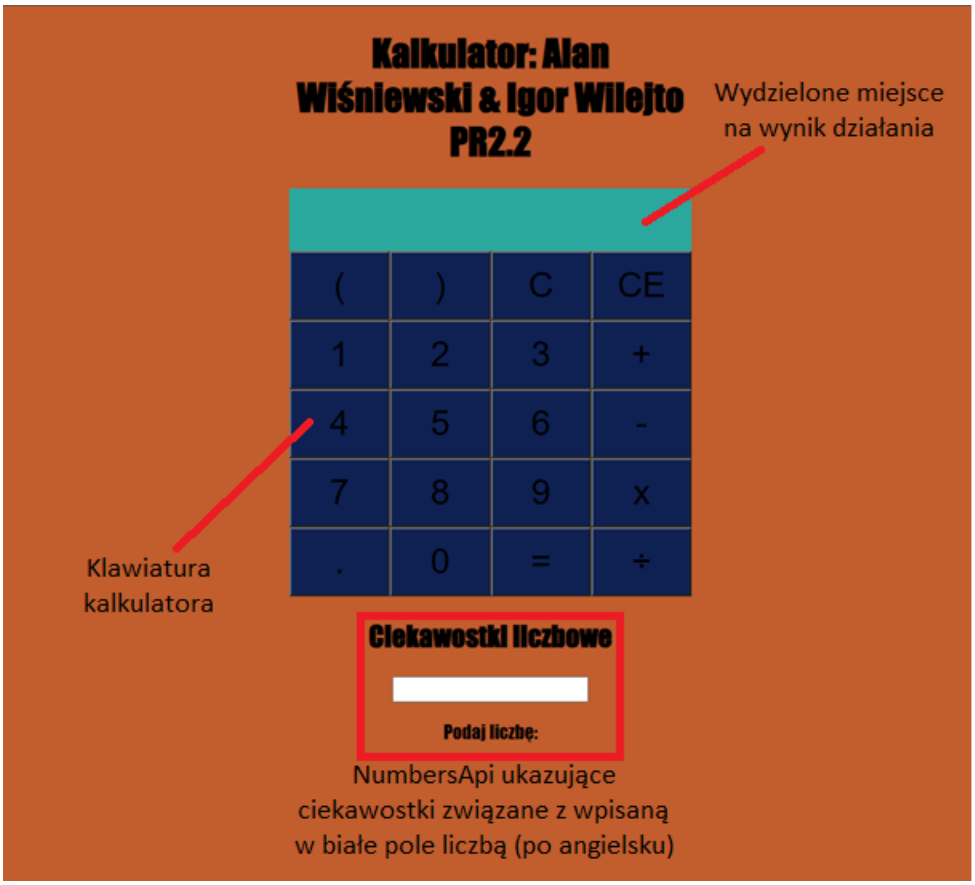
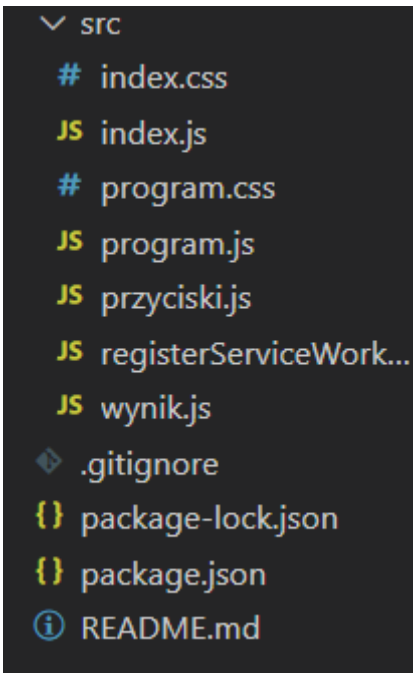
Kalkulator REACT.js

Autorzy:
Alan Wiśniewski
Igor Wilejto

Działanie aplikacji



Zawartość głównego katalogu



Index.css – Zawiera instrukcje stylistyczne dla całego body naszego programu.
index.js – Jest to punkt wejściowy dla komponentów programu.
program.css - Zawiera instrukcje stylistyczne dla programu.
program.js – Serce i mózg programu. Zawiera kluczowe instrukcje programistyczne, oraz wykorzystuje informacje zawarte w innych plikach.
przyciski.js – Zdefiniowane przyciski kalkulatora.
registerServiceWorker.js – API, które pomaga buforować zasoby, co przyspiesza nam działanie aplikacji. Również bardzo ważne w tym API jest możliwość uzyskania dostępu offline podczas produkcji aplikacji.
wynik.js – instrukcja programistyczna odpowiedzialna za ukazywanie wyniku na ekranie.

Kod

Wynik.js

```
wynik.js > default
import React, {Component} from 'react';
//niniejszy plik eksportuje wynik do głównego pliku "program.js"
class Wynik extends Component {


    render() {
        let {wynik} = this.props;
        return (
            <div className="wynik">
                <p>{wynik}</p>
            </div>
    )
        ;
    }
}


export default Wynik;
```

Render() odpowiedzialny jest za wyświetlenie co następuje:
„let” wykorzystujemy zamiast const, ponieważ wartość w zmiennej „wynik” jest bezpośrednio przez nas zmieniana.
Return zwraca nam wynik w wyglądzie przygotowanym w pliku program.css. Tam instrukcje stylistyczne również zostały zdefiniowane pod nazwą klasy „wynik”.
Tak przygotowany komponent został wyeksportowany jako **Wynik.**

Przyciski.js

```
r() {
eturn (
    <div className="przyciski">
        <button name="(" onClick={e => this.props.onClick(e.target.name)}>(</button>
        <button name=")" onClick={e => this.props.onClick(e.target.name)}>)</button>
        <button name="C" onClick={e => this.props.onClick(e.target.name)}>C</button>
        <button name="CE" onClick={e => this.props.onClick(e.target.name)}>CE</button><br


        <button name="1" onClick={e => this.props.onClick(e.target.name)}>1</button>
        <button name="2" onClick={e => this.props.onClick(e.target.name)}>2</button>
        <button name="3" onClick={e => this.props.onClick(e.target.name)}>3</button>
        <button name="+" onClick={e => this.props.onClick(e.target.name)}>+</button><br/>


        <button name="4" onClick={e => this.props.onClick(e.target.name)}>4</button>
        <button name="5" onClick={e => this.props.onClick(e.target.name)}>5</button>
        <button name="6" onClick={e => this.props.onClick(e.target.name)}>6</button>
        <button name="-" onClick={e => this.props.onClick(e.target.name)}>-</button><br/>

        <button name="7" onClick={e => this.props.onClick(e.target.name)}>7</button>
        <button name="8" onClick={e => this.props.onClick(e.target.name)}>8</button>
        <button name="9" onClick={e => this.props.onClick(e.target.name)}>9</button>
        <button name="*" onClick={e => this.props.onClick(e.target.name)}>x</button><br/>


        <button name="." onClick={e => this.props.onClick(e.target.name)}>.</button>
        <button name="0" onClick={e => this.props.onClick(e.target.name)}>0</button>
        <button name="=" onClick={e => this.props.onClick(e.target.name)}>=</button>
        <button name="/" onClick={e => this.props.onClick(e.target.name)}>÷</button><br/>
    </div>
;
```

Tutaj zdefiniowane zostały przyciski będące naszym bezpośrednim narzędziem podczas wykonywania obliczeń na kalkulatorze. Z tej racji event kliknięcia musi zostać przekazany do „rodzica” (czyli jaki przycisk został kliknięty) dlatego przy każdym przycisku używamy *this.props.OnClick* oraz przepuszczamy *e.target.name* jako argument.

Program.js

```
import React, { Component } from 'react';
import './program.css';
import Wynik from './wynik';
import Klawiatura from "./przyciski";

class App extends Component {
    constructor(){
        super();

        this.state = {
            wynik: "",
            text: "Podaj liczbę: ",
            error: ""
        };
    }

    onClick = przycisk => {

        if(przycisk === "="){
            this.licz()
        }

        else if(przycisk === "C"){
            this.resetowanie()
        }
        else if(przycisk === "CE"){
            this.spacja()
        }

        else {
            this.setState({
                wynik:
             this.state.wynik + przycisk
            })
        }
    };
```

```
resetowanie = () => {
    this.setState({
        wynik: ""
    })
};


licz = () => {
    try {
        this.setState({
            wynik:
         (eval(this.state.wynik) || "" ) + ""
        })
    } catch (e) {
        this.setState({
            wynik:
         "Błąd!"
        })

    }
};

cofnięcie = () => {
    this.setState({
        wynik:
     this.state.wynik.slice(0, -1)
    })
};
componentDidMount = (e) => {
    const value = this.refs.number.value;
    console.log(value);
    fetch(`http://numbersapi.com/${value}/year?json`)
    .then(res => {
        if(res.ok) {
            return res
        }
        throw Error(res.status)
        })
```

```
        fetch(`http://numbersapi.com/${value}/year?json`)
        .then(res => {
            if(res.ok) {
                return res
            }
            throw Error(res.status)
            })

        .then(res =>  res.json())
        .then(data => this.setState({
            text: "W tym roku: " + data.text
        }))
        .catch(err => console.log(err))
}

    render() {
        return (
            <div>
                <div className="kalkulator_css">
                    <h1>Kalkulator: Alan Wiśniewski & Igor Wilejto PR2.2</h1>
                    <Wynik wynik={this.state.wynik}/>
                    <Klawiatura onClick={this.onClick}/>
                </div>
                <h2> Ciekawostki liczbowe </h2>
                <div className="ciekawostki">
                <input onChange={this.componentDidMount} type="text" ref="number" />
                <p> {this.state.text} </p>
                </div>
            </div>

        );

    }

}

export default App;
```

Program,js to jest wspomniany rodzic naszych komponentów (przyciski,js oraz wynik.js).

Kod dołącza wszystkie dzieci oraz renderuje komponenty, które powinny zostać wyświetlone aby uzyskać pożądany efekt.
Zaczynamy od dołączenia „wynik” jako zmienna w this.state, co pozwoli nam na manipulowanie tym co będzie wyświetlane na ekranie. Również mamy zmienną „text”, która została wykorzystana w API programu.
Skonstruowane zostały cztery główne cechy programu:

licz() – Oblicza wynik. Aktywuje się po naciśnięciu przycisku „=”. Wszystko zostało zamknięte w pętli, okraszone funkcją wykrywania błędów (czyli gdy zostanie wprowadzony niepożądany znak lub składnia działania będzie niespójna).

reset() – Czyści cały ekran wyniku. Aktywuje się po wciśnięciu przycisku „C”.
cofnięcie() – Usuwa pojedynczy znak na ekranie wyniku. Aktywuje się po wciśnięciu przycisku „CE”. Funkcja *slice* została w tym przypadku wykorzystana do osiągnięcia zamierzonego efektu.
onClick = przycisk – Tutaj składamy w spójną całość wszystko to, co stworzyliśmy wcześniej i przypisujemy konkretnym metodom odpowiadające im przyciski kalkulatora.

Do połączenia się z api wykorzystaliśmy funkcję *fetch*. Składnia API wygląda następująco:

http://numbersapi.com/number/type

Number to dowolna liczba, na podstawie której zostanie wyświetlona ciekawostka. *Type* to rodzaj tej ciekawostki. Wyróżniamy cztery rodzaje: trivia, math, date, year.
My wykorzystaliśmy year, aby były to ciekawostki związane z latami, natomiast liczba w naszym zamyśle miała być wprowadzona przez użytkownika, dlatego umieszczamy w tym miejscu *value*.
Po wydobyciu danych z api, poddajemy je weryfikacji. Jeżeli wynik(res) jest poprawny, zwraca nam ten wynik. W innym wypadku wyskakuje błąd. Wykorzystujemy metodę json() aby wyodrębnić potrzebne informacje.
Następnie te dane zostają przekazane dalej. Tutaj użyliśmy nazwy *data* by następnie za pomocą metody *setState()* wpłynąć na zawartość zmiennej *text* i wyświetlić pożądane informacje w wizualnej części aplikacji.

Na końcu wszystko zostało przekazane do metody render(), czyli wyświetlamy wynik z wykorzystaniem odwołania do komponentu, który wyeksportowaliśmy pod nazwą *Wynik.*
Również przyciski z odwołaniem do komponentu *Klawiatura* zostały umieszczone w wizualnej części aplikacji.
Na końcu stworzone zostało pole input, które zmienia się w oparciu o instrukcje połączoną z API i wyświetla ciekawostki, które wywołuje użytkownik.

Program. Css

```css
.wynik {
    background-color: rgb(43, 168, 158);
    width: 100%;
    height: 55px;
  }

  .wynik p {
    font-size: 45px;
    margin: 10px;

  }

  .kalkulator_css {
    max-width: 350px;
    margin: auto;
    text-align: center;
  }

  .przyciski {
    display: block;
  }

  button {
    width: 25%;
    height: 60px;
    font-size: 30px;
    background-color: rgb(15, 33, 83);
  }
  button:hover {
    background-color: rgb(26, 54, 131);
  }
  h2 {
    text-align:center;
  }

  .ciekawostki{
    text-align:center;
  }
```

Index.css

Zakończenie

Mamy nadzieję, że aplikacja spełnia pańskie wymagania. Włożyliśmy w nią sporo pracy i będąc osobami nieobeznanymi z frameworkiem React, sądzimy, iż wykonana praca wygląda zadowalająco :).