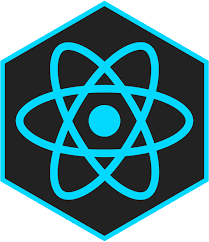
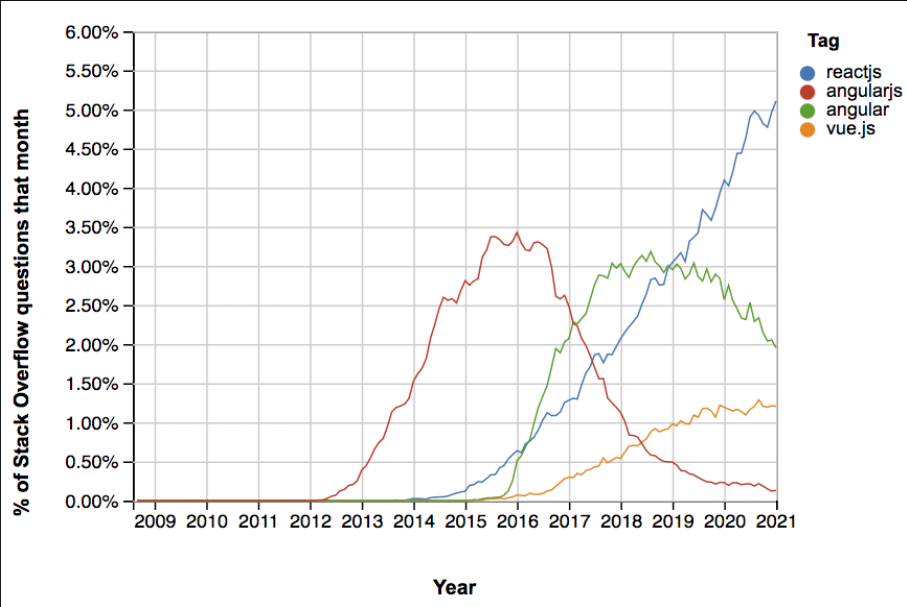
****

**Szkolenie 1 - React.js**

**Wprowadzenie i pierwszy komponent**

**Wprowadzenie**

Witam Cię bardzo serdecznie w pierwszej części kursu z podstaw React. Gratuluje Ci odwagi i chęci prowadzenia ciągłego rozwoju poprzez naukę nowych technologii. To pierwszy krok w kierunku realizacji Twoich ambicji. Na świecie cały czas wzrasta zapotrzebowanie na programistów, nie tylko front-endu, ale i back-endu. Uwielbiam frontend, ponieważ czuję, że jestem wtedy bardzo blisko naszego końcowego odbiorcy – użytkownika. Bardzo cieszę się, kiedy widzę, że moje aplikacje są używane w zakładach pracy, w telefonach, na komputerach. Dzięki wytworzonemu przez nas oprogramowaniu wzrasta przyjemność użytkowania aplikacji, czasem nawet optymalizujemy niektóre procesy, zmniejszając wpływ człowieka na środowisko.   
  
React jest bardzo wydajnym frameworkiem, służy głównie do tworzenia elastycznych i rozbudowanych aplikacji internetowych. Za pomocą React-a możemy również tworzyć aplikacje **PWA** (Progressive Web App – to taka aplikacja, której zawartość jest zapisywana w pamięci urządzenia, dzięki temu ponowne wejście na stronę może odbyć się w trybie offline). Za pomocą React Native możemy natomiast tworzyć aplikacje natywne na smartfony i tablety, a łącząc biblioteki Electron i React, możemy również stworzyć aplikacje desktopowe. Przez to React jest aktualnie najpopularniejszym frameworkiem, więc z jego znajomością, jest o wiele łatwiej podołać wymaganiom rynkowym i znaleźć wymarzoną pracę.



Popularność React.js na tle innych frameworków

**Źródło:** [Front-end frameworks popularity (React, Vue and Angular)](https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190)

Na powyższym wykresie widać, jak bardzo popularny jest React, ma on dużą społeczność programistów wspierających innych i odpowiadających na pytania zadawanie na [Stack Overflow](https://stackoverflow.com/), z którego na pewno często będziesz korzystać.

W przeciwieństwie do Angular-a, React daje nam szeroki wachlarz możliwości. Możemy użyć go w wielu miejscach na różne sposoby, łącząc go z wieloma rozwiązaniami. Wystarczy tylko mieć otwarty umysł.   
  
W tej części kursu poznasz podstawowe koncepty Reacta, nauczysz się łączyć ze sobą różne zagadnienia i tworzyć **komponenty**.

**React od strony nieco bardziej technicznej…**

Na początku chciałbym Ci krótko opowiedzieć, czym jest React. React jest biblioteką do budowania interfejsów w JavaScript, stworzoną przez programistów Facebooka. Zdobył ogromną popularność, pisze się w nim bardzo przyjemnie. Koncept Reacta opiera się na znanych wzorcach, jak np. jednokierunkowy przepływ danych i budowanie aplikacji w oparciu o komponenty.

W internecie nadal trwa spór, czy React jest frameworkiem, czy biblioteką. Dla mnie osobiście nie ma to znaczenia - tutaj, aby wykonać zapytanie do API (Application Programming Interface – interfejs webowy zwracający odpowiednie dane dla zadanego zapytania), używamy zewnętrznych paczek, których jest ponad kilkanaście tysięcy! (dostępne paczki możesz znaleźć tutaj:<https://www.npmjs.com/>). Ogromna społeczność pomoże wybrać odpowiednie rozwiązania. Dzięki temu, że korzystamy z wielu rozwiązań stajemy się bardzo elastyczni. W dodatku nie zamykamy się na nowe rozwiązania i programujemy często w czystym JS.

Przewagą Reacta nad innymi bibliotekami/frameworkami jest:

* **VirtualDOM** - za jego pomocą renderowanie w przeglądarce jest szybsze niż kiedykolwiek wcześniej. React dba o to, żeby aktualizować tylko te fragmenty drzewa strony, które się zmieniły.
* **Deklaratywny kod** - dzięki temu, kod staje się bardziej przejrzysty i łatwiejszy do debugowania. Więcej o deklaratywnym programowaniu możesz nauczyć się z naszych kursów JS-a lub Python-a.
* **Duża społeczność** - łatwo jest znaleźć pomoc w jakimkolwiek problemie. Jeżeli natrafiłeś na ścianę istnieje ogromna szansa, że już ktoś miał podobny problem.

**Create-react-app**

Najwygodniejszym rozwiązaniem do tworzenia rozwiązań React-owych, jest zastosowanie zintegrowanego zestawu narzędzi. Create-React-App jest skryptem tworzącym szablon z dogodnym środowiskiem dla jednostronicowej aplikacji w React. Środowisko pracy stworzone przez Create-React-App umożliwi stosowanie najnowszych funkcjonalności języka JS, zoptymalizuje Twój kod i usprawni Twoją pracę. Do tego zagadnienia wrócimy, gdy przystąpimy już do tworzenia pierwszej aplikacji.

**Zanim zaczniesz…**

Swój rozwój i naukę musimy oprzeć na solidnych fundamentach, trochę tak jak przy budowie domu. Dlatego w poniższym akapicie przedstawię Ci w skrócie, z jakimi zagadnieniami powinieneś się zapoznać przed przystąpieniem do pisania React-owego kodu. Tak, aby nauka była łatwa, prosta i przyjemna.

Bez tych podstaw ani rusz! W dodatku w celu obiektywnego określenia, czy podołasz i czy jesteś gotowy na pracę z React-em, poproszę Cię o rozwiązanie kilku zadań.

**Wymóg nr 1**

HTML5 – głównie to praca ze znacznikami, tagami, atrybutami. HTML jest naszym szkieletem stron webowych, jako programista React-a, strukturę stron będziesz tworzył właśnie przy użyciu HTML-a.   
  
Dobrym ćwiczeniem sprawdzającym Twoje umiejętności z HTML-a, ale również ugruntowującym poznane zagadnienia, może być zaprojektowanie struktury HTML składającej się z:

* menu, będącego linkami do poszczególnych sekcji strony,
* sekcji „hero”
* sekcji zawierająca artykuł z obrazkiem,
* sekcji zawierająca galerię,
* footer-a

**Wymóg nr 2**

CSS3 – tutaj warto znać pozycjonowanie (flex, grid), selektory, pseudoklasy. Warto wiedzieć jak oblicza się specyficzność selektorów, znać model pudełkowy, gradienty, sposoby zapisu kolorów, grafiki, zasady dobierania palety kolorystycznej, dodawanie fontów, tworzenie przejść, animacji, RWD oraz zmienne CSS.

W celu sprawdzenia samego siebie, możesz dodać do szkieletu strony z poprzedniego ćwiczenia poniższe wytyczne:

* Wykorzystaj pozycjonowanie oraz flex do ostylowanie menu sekcji hero oraz sekcji artykułu z obrazem,
* Grid idealnie nadaje się do sekcji zawierającej galerię,
* Footer możesz stylizować wedle uznania

Możesz również zagrać w grę [CSS Diner - Where we feast on CSS Selectors!](https://flukeout.github.io) aby zapoznać się lepiej z selektorami CSS, aby przypomnieć sobie flexa polecam grę flexbox froggy [Gra wprowadzająca do CSS flexbox"](https://flexboxfroggy.com/#pl) a dla grida grę grid-garden [Grid Garden - A game for learning CSS grid](https://cssgridgarden.com/#pl)

**Wymóg nr 3**

JavaScript.

Upewnij się, że wiesz, jak wykorzystywać zmienne (let, const, var), typy danych, operatory, funkcje, instrukcje warunkowe, ternary operator, typy referencyjne, arrow functions, tablice, właściwości i metody wykorzystywane do pracy na tablicach, pętle for, while do while, for of, zagnieżdżanie pętli, praca na tablicach za pomocą pętli, funkcje wyższego rzędu (tzw. High Order Functions), metody wykorzystywane do iterowania po tablicach, obiekty, dostęp do właściwości i metod w obiekcie, słowo kluczowe this, klasy, metody klasy, dziedziczenie, metody statyczne, interakcje z DOM poprzez JS, reagowanie na zdarzenia użytkownika, asynchroniczność w JS.

Ilość zagadnień, z jaką należy się zapoznać, może wygląda przerażająco, ale pamiętajmy, że JavaScript jest pełnoprawnym językiem programowania. Warto również zapoznać się z zagadnieniami dotyczącymi dostępności, aby nasza aplikacja była przyjazna dla osób niepełnosprawnych.

CSS3 oraz JS to zagadnienia, przy których powinno się zostać trochę dłużej. Jak to mówi stare powiedzenie - od przybytku głowa nie boli, a wartość rynkowa programisty drążącego temat bardziej i bardziej, staje się coraz większa.

**Edytor kodu i wymagane narzędzia**

Poniżej znajdziesz listę narzędzi i paczek, które musisz zainstalować, aby móc zacząć programować w React:

1. **System kontroli wersji GIT**

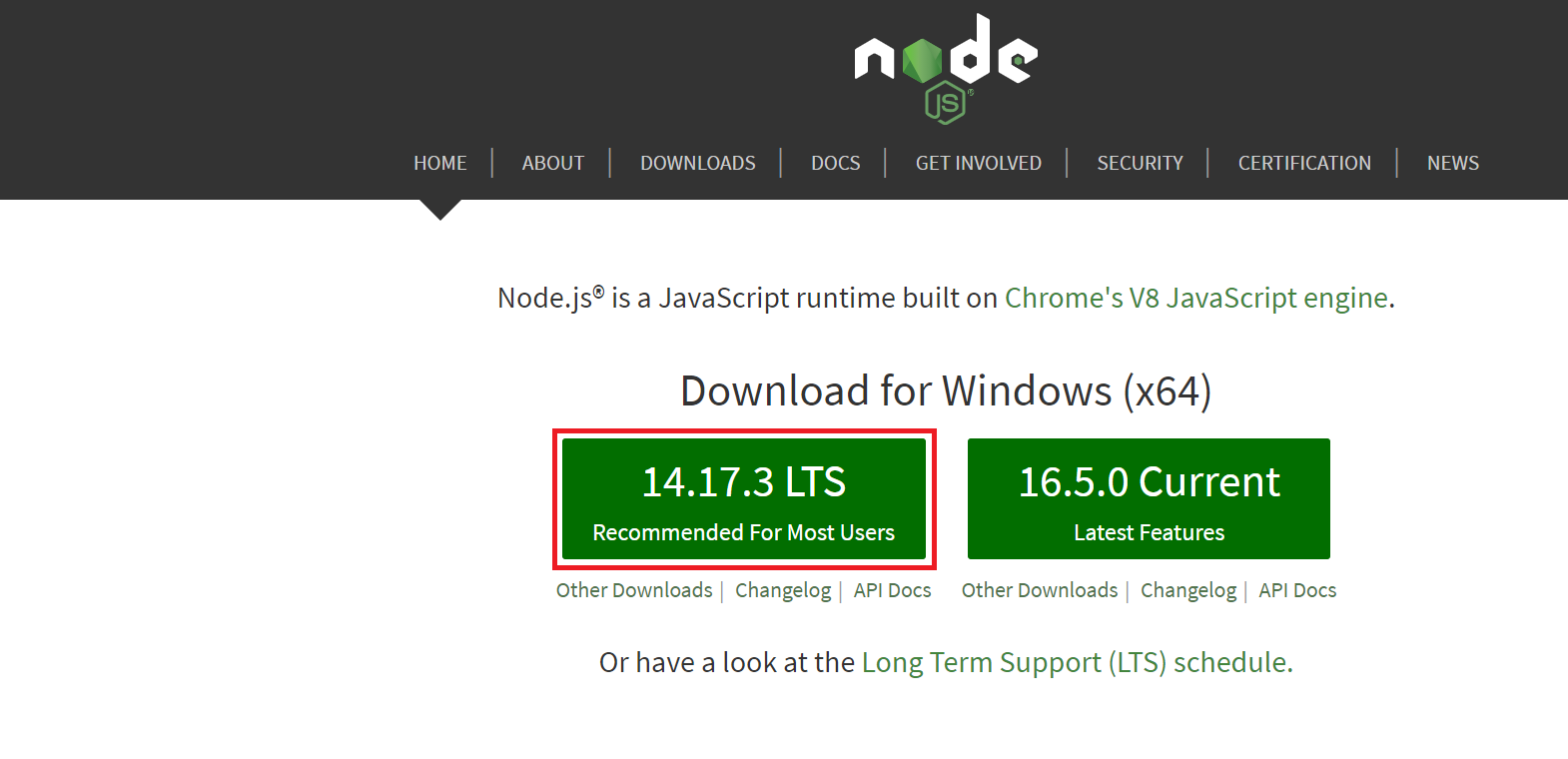
GIT to narzędzie używane przez każdego programistę, służy do współpracy pomiędzy developerami oraz do wersjonowania kodu. GIT pobierzesz ze strony <https://git-scm.com/downloads>.

Jeżeli nie wiesz jeszcze, jak z niego korzystać, to wróć do naszego szkolenie z GIT-a, w którym dogłębnie opisaliśmy każde zagadnienie.

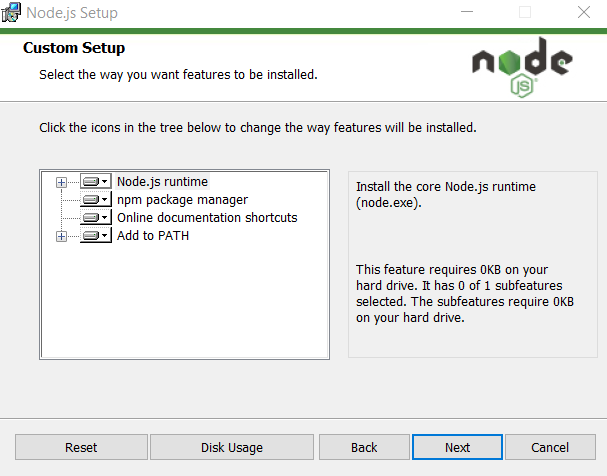
Po instalacji możemy automatycznie używać Git Basha (dołączonego do pakietu instalacyjnego GIT).

1. **Wieloplatformowe środowisko uruchomieniowe Node**

Możesz je pobrać stąd: [Node.js](https://nodejs.org/en/). Najlepiej, abyś wybrał wersję LTS (Long Term Support).



Wybór odpowiedniej wersji Node

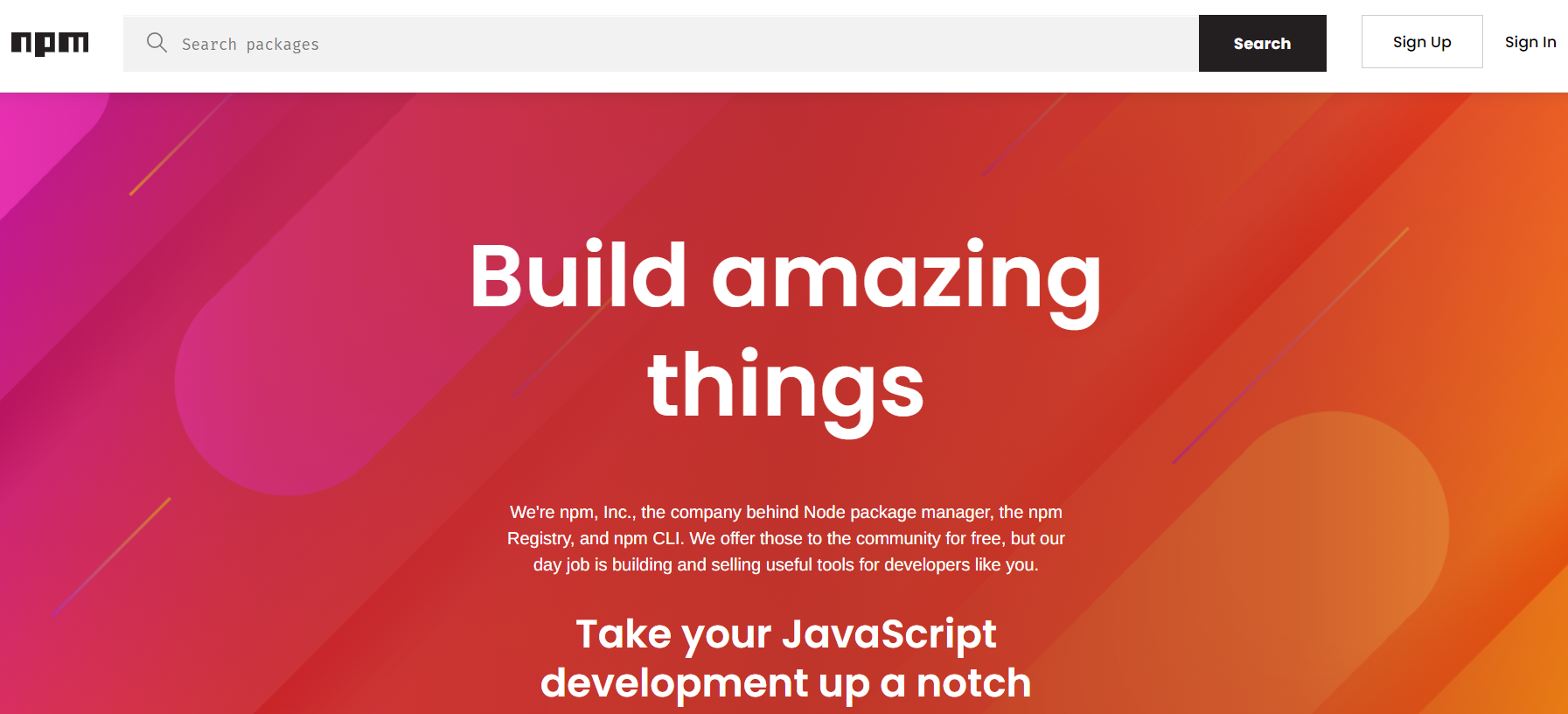


Uruchomienie instalatora Node

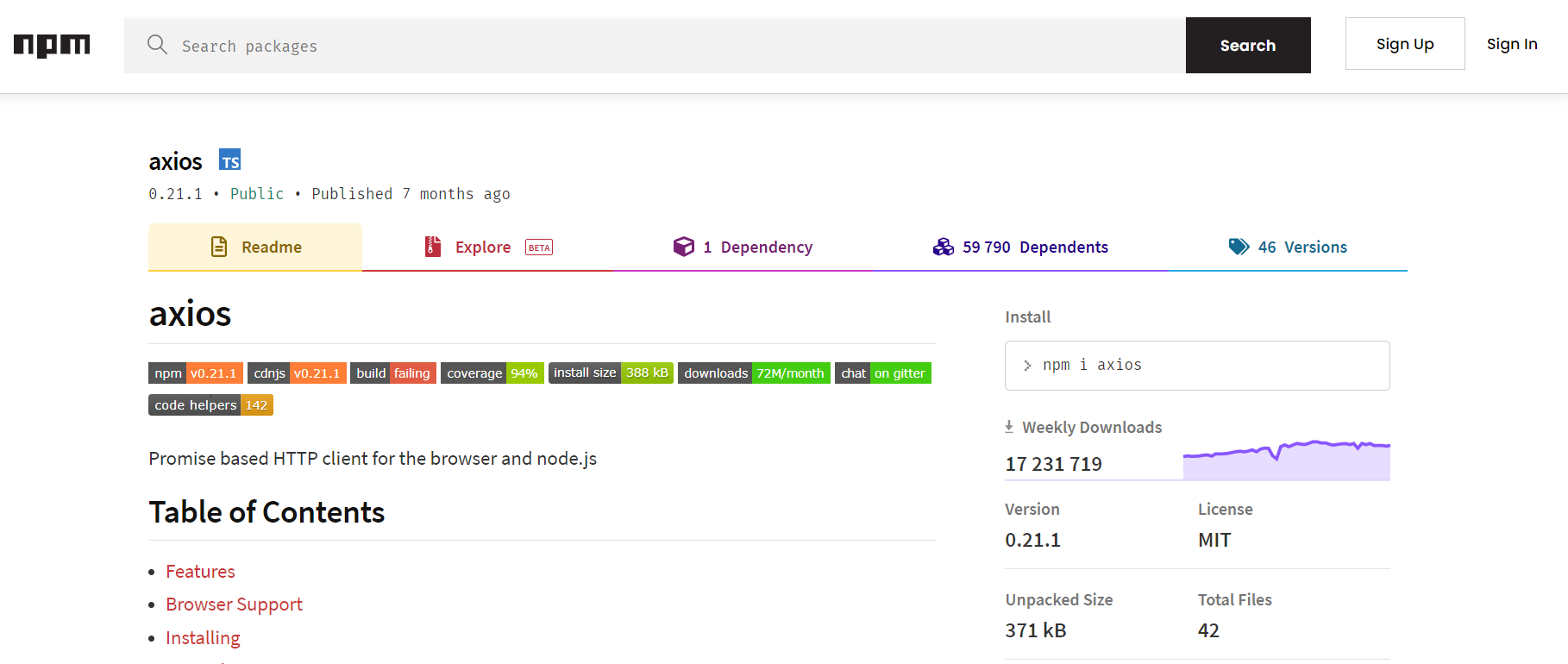
Proces instalacji i jest łatwy i intuicyjny, tak więc nie będzie konieczne, abym prowadził Cię przez niego krok po kroku.

**3. Menedżer pakietów Npm**

Domyślny manager pakietów dla środowiska Node.js. Jest on używany do zarządzania warstwą front-end aplikacji. Jest to więc repozytorium pakietów. Nazwa NPM pochodzi od firmy, która odpowiada za jego stworzenie. Innym przykładem menedżera pakietów może być yarn, my jednak nie będziemy z niego korzystali.   
  
Całe repozytorium NPM dostępne jest do pobrania pod adresem: [npm](https://www.npmjs.com)



Na stronie głównej znajduje się wyszukiwarka, dzięki której możemy znaleźć naszą paczkę i zapoznać się z jej statystykami. Tak wygląda przykładowa podstrona dla paczki axios służącej do komunikacji z API.



Statystyka dla axios

**4. Edytor kodu VSCode**

Środowiskiem, które będę używał do edycji kodu podczas tego kursu, będzie **Visual Studio Code**, które jest bardzo elastyczne dzięki możliwości personalizacji produktu pod swoje potrzeby. Producenci bowiem zapewniają nam niezliczoną ilość wtyczek, które mogą efektywnie usprawniać naszą pracę z środowiskiem.   
  
VSCode możemy pobrać z tej strony: [Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com)

**Wykorzystywane wtyczki do VSCode**

Poniżej prezentuję skróconą listę moich wtyczek wraz z krótkim opisem:

***bradgashler.htmltagwrap*** – ułatwia pracę z HTMLem pozwala owrapowywać zaznaczone elementy.

***CoenraadS.bracket-pair-colorizer-2*** – zwiększa czytelność kodu

***dbaeumer.vscode-eslint*** – integruje ESLinta z VSCode, służy do statycznej analizy kodu

***dsznajder.es7-react-js-snippets*** – zbiór bardzo przydatnych snippetów do szybszego pisania kodu

***esbenp.prettier-vscode*** – formater kodu

***ritwickdey.LiveServer*** – live server pozwala na udostępnianie swojego kodu w sesji innym programistom

***steoates.autoimport*** – pomaga podczas importowania paczek npm, dba o dobrą ścieżkę importu

***streetsidesoftware.code-spell-checker*** - sprawdza pisownię, pomaga w unikaniu literówek

***thedavej.night-owl-black*** – mój motyw kolorów

***VisualStudioExptTeam.vscodeintellicode*** – AI podpowiadające składnię

***vscode-icons-team.vscode-icons*** – ikony VSCode

***wix.vscode-import-cost*** – wtyczka ważna ze względów wydajnościowych. Im mniejsza aplikacja tym szybciej się ściąga i lepiej działa, wtyczka ta pomaga określić, jaki jest koszt importu danej paczki.  
  
  
**Budowa pierwszej aplikacji**

To tyle, jeśli chodzi o konfigurację naszego środowiska pracy oraz instalację wszystkich potrzebnych dependencji! Teraz czas na chwalebny moment - stwórzmy pierwszą React-ową aplikację.

**Pierwszy komponent**

Zanim przejdziemy do omówienia, jak zbudować pierwszy komponent w React, warto by odpowiedzieć sobie na pytanie, co kryje się pod pojęciem “komponent”.

Z czysto technicznego punktu widzenia, komponenty to nic innego jak funkcje lub klasy. Opcjonalnie przyjmują dane wejściowe w postaci obiektu props (o którym więcej za chwilę), a ich danymi wyjściowymi są elementy React.

Aby stworzyć pierwszy komponent, skorzystamy z pomocy Git Basha. Po uruchomieniu tego terminala, przechodzimy za pomocą polecenia **cd,** do folderu, w którym będziemy tworzyli aplikację, np. **e:/kurs**

|  |
| --- |
| cd e:/kurs |

W tym folderze stworzymy za pomocą skryptu **create-react-app** swoją aplikację.

Najpierw zainstalujemy create-react-app globalnie (przełącznik **-g**).

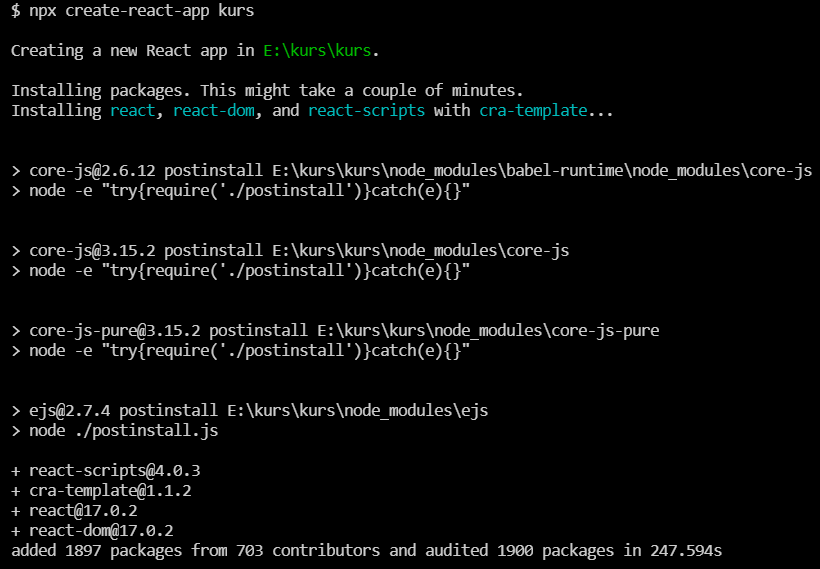
|  |
| --- |
| npm install -g create-react-app |

**UWAGA:**

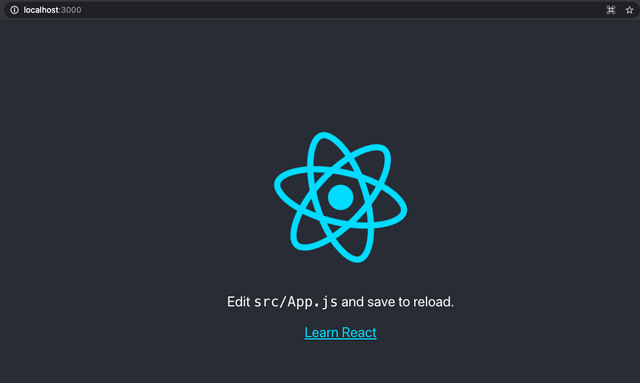
Flaga -g umożliwia globalną instalację paczki i dodanie jej do zmiennych środowiskowych systemu. Dzięki temu będzie ona dostępna z poziomu dowolnego katalogu projektowego w terminalu.

Następnie tworzymy nowy projekt React-owy poleceniem:

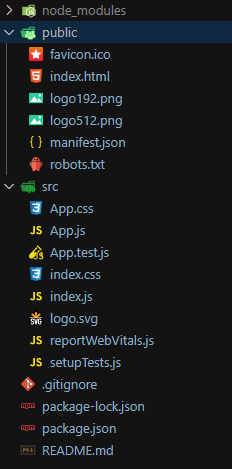
|  |
| --- |
| npx create-react-app <nazwa\_aplikacji> |

Utworzenie nowej aplikacji React

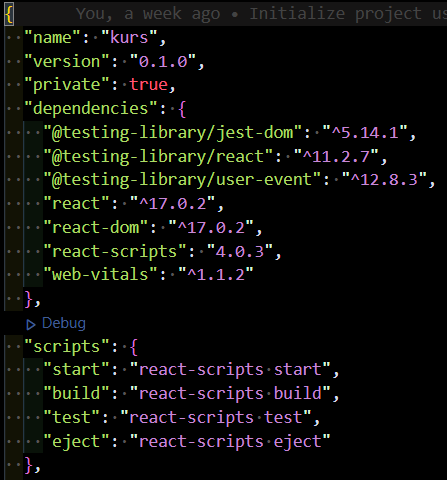
Następnie wchodzimy i uruchamiamy projekt za pomocą poleceń: cd kurs, a następnie npm start.   
  
Wówczas **localhost:3000** powinna otworzyć się nasza pierwsza strona napisana w React.js.

Uruchomienie pierwszej aplikacji React

Przyjrzyjmy się strukturze utworzonego projektu w VSCode

  
Struktura projektowa aplikacji kurs

Mamy tutaj folder **node\_modules**, w którym znajdują się nasze paczki, folder **public** w którym umieszczone są pliki zbudowanej aplikacji wraz z obrazkami, **robots.txt** służy do indeksowania strony przez roboty firmy Google ([Robots.txt Introduction & Guide | Google Search Central](https://developers.google.com/search/docs/advanced/robots/intro)), a **manifest.json** jest opisem aplikacji.   
  
Teraz zajrzymy do folderu src/. To tutaj znajduje się source (źródło) naszej aplikacji. W tym miejscu będziemy tworzyć komponenty.   
  
W naszym szablonie znajduje się również pliki **README.md** służący do opisu repozytorium, **.gitignore** który jest plikiem konfiguracyjnym gita oraz **package.json**, w której znajduje się konfiguracja oraz wersje zainstalowanych paczek npm. Jak widać w pliku tym umieszczona jest paczka **react**, **react-dom**, **react-scripts**.

  
Plik package.json

Zajrzyj do pliku index.html w folderze public/. Mamy tutaj div-a o **id=root.** Jest to nasz korzeń aplikacji, to tutaj montowana jest cała aplikacja do HTMLa.

<body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div *id*="root"></div>

W pliku **index.js** znajduje się funkcja renderująca, i montująca komponenty React-owe w naszej aplikacji.

Tagi pisane z dużej litery to właśnie komponenty Reactowe (np. <App />). W pliku index.js montowany jest w StrictMode komponent App, a za pomocą funkcji **ReactDOM.render** montowany w **index.html** w divie o id=root.

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

Zajrzyjmy teraz do pliku App.js.

import logo from './logo.svg'

import "./App.css";

function App() {

*return* (

    <div *className*="App">

      <header *className*="App-header">

        <img *src*={logo} *className*="App-logo" *alt*="logo"/>

        <p>

          Edit <code>src/App.js</code> and save to reload.

        </p>

        <a

*className*="App-link"

*href*="https://reactjs.org"

*target*="\_blank"

*rel*="noopener noreferrer">

          Learn React

        </a>

      </header>

    </div>

  );

}

export default App;

Jak widzisz znajduje się tutaj funkcja (o komponentach klasowych opowiem innym razem), zwracająca za pomocą return kod HTML. W komponencie App w przyszłości umieścimy inne komponenty wraz z routingiem.   
  
Na ten moment wyrzucimy z return cały obecny HTML i umieścimy w nim customowany kod (proste wyświetlanie dwóch akapitów tekstu). Powinieneś otrzymać poniższy efekt:

import "./App.css";

function App() {

*return* (

    <div>

      <h1>React is great!</h1>

      <p>React is beautiful</p>

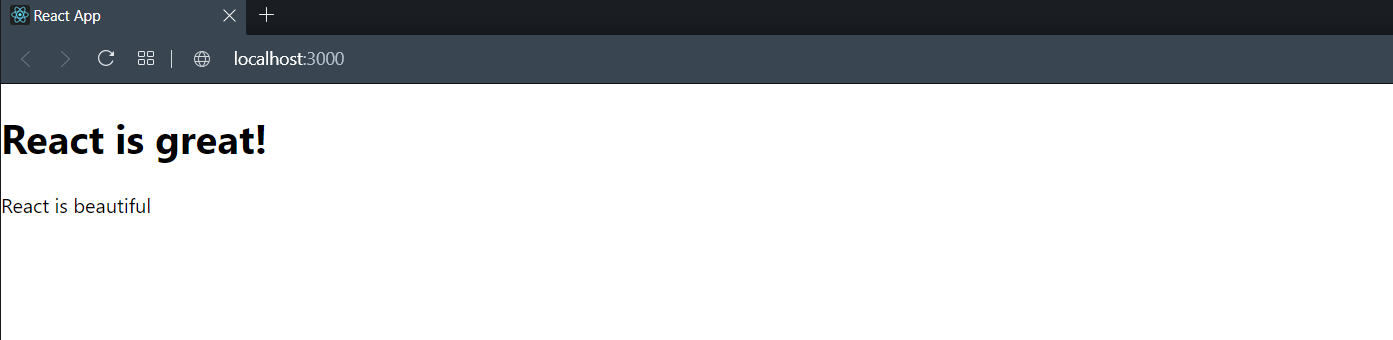
    </div>

  );

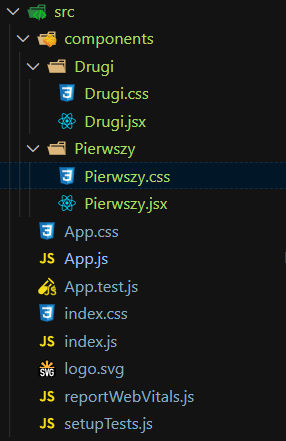
}

export default App;

W przeglądarce pod adresem localhost:3000 możesz zobaczyć jak aplikacja się zmieniła (pamiętaj, aby aplikacja była uruchomiona wcześniej wykonanym poleceniem npm start). U mnie wygląda w ten sposób.



Stwórzmy teraz dwa nowe komponenty. Tworzymy folder components/, wewnątrz niego katalogi Pierwszy i Drugi, a w nich Pierwszy.jsx, Pierwszy.css oraz Drugi.jsx oraz Drugi.css.

  
Struktura katalogu po zmianach

Stwórzmy kod Pierwszego oraz Drugiego komponentu.

import React from 'react'

const First = () => {

*return* (

      <div>

        Pierwszy!

      </div>

    );

}

export default First;

Na drugim komponencie pokażę przykład jak używać plików CSS wewnątrz komponentu. Skorzystamy z prostego kodu CSS umieszczonego wewnątrz Drugi.css.

.*red* {

    color: red;

}

Importujemy kod css dla Drugiego komponentu (Drugi.jsx), a następnie za pomocą atrybutu className przypisujemy do elementu klasę utworzonego atrybutu CSS.

import React from 'react'

import "./Second.css"

const Second = () => {

*return* (

        <div *className*="red">

            Drugi

        </div>

    )

}

export default Second

Teraz musimy jeszcze zaimportować nasze komponenty do App.js (nie jest to wzorcowy sposób importu, ale na chwilę obecną nam wystarczy).

import Second from "./components/Drugi/Drugi";

import First from "./components/Pierwszy/Pierwszy";

import "./App.css";

function App() {

*return* (

    <div>

      <h1>React is great!</h1>

      <p>React is beautiful</p>

      <First></First>

      <Second/>

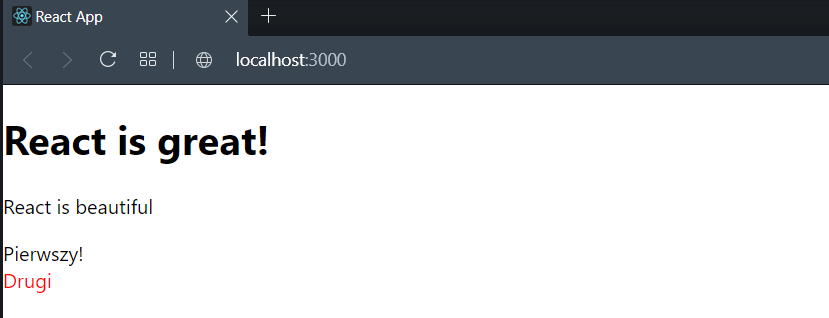
    </div>

  );

}

export default App;

Oto efekt jaki uzyskaliśmy w przeglądarce. Komponent drugi jest czerwony.



Komponentów możemy używać kilka razy i każdy z nich jest osobną instancją.

import Second from "./components/Drugi/Drugi";

import First from "./components/Pierwszy/Pierwszy";

import "./App.css";

function App() {

*return* (

    <div>

      <h1>React is great!</h1>

      <p>React is beautiful</p>

      <First></First>

      <Second />

      <First/>

      <Second />

      <Second />

      <Second />

    </div>

  );

}

export default App;

Zauważ, że komponenty możemy używać tak samo jak tagi HTML.

Aby przećwiczyć importowanie komponentów i ich tworzenie sprawdź jaki efekt otrzymamy w przeglądarce po wykorzystaniu kodu z powyższego szkolenia

|  |
| --- |
| **Czym różni się rozszerzenie js od jsx?**  JS jest rozszerzeniem JavaScript, a JSX łączy ze sobą kod JS-a z HTML-em. Z plików .jsx korzystamy wtedy, gdy chcemy zdefiniować, jak ma wyglądać UI. Jeśli używamy TypeScriptu będziemy korzystali z rozszerzeń .ts oraz .tsx. Ja osobiście używam rozszerzenia .jsx, bo moje wtyczki lepiej z nim współgrają, ale dla Reacta nie ma znaczenia czy używasz rozszerzenia js czy jsx. |

**Argumenty komponentu propsy**

Obecnie posiadamy trzy komponenty **App**, **First**(czarny tekst) oraz **Second** (czerwony tekst). Jak już zdążyłeś zauważyć, instancje komponentów możemy wykorzystywać wielokrotnie i umieszczać je w różnych miejscach kodu. Z komponentami wiąże się jeszcze jedno dość ważne zagadnienie - **propsy**.

Propsy są to elementy (argumenty), które możemy przekazać do każdego komponentu. Ich ilość definiuje programista, mogą to być dane do obliczeń, tekst lub zmienne **zmieniającego** nasz generyczny komponent.

Wykorzystamy tutaj komponent Pierwszy do wyświetlania wyników działania, a komponent Drugi do wyświetlania dostosowanego tekstu.

Zmieńmy na początek komponent Pierwszy.

import React from 'react'

const First = ({a,b}) => {

*return* (

      <div>

        <p>{a+b}</p>

        <p>{a\*b}</p>

      </div>

    );

}

export default Pierwszy

Propsy definiujemy tak samo jak parametry funkcji, dodatkowo użyłem tutaj destrukturyzacji (jeżeli nie pamiętasz z JS, czym jest destrukturyzacja, zajrzyj tutaj: [Destructuring assignment - JavaScript | MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment) ).

Następnie zmieniamy komponent drugi.

import React from "react";

import "./Second.css";

const Second = ({ tekst }) => {

*return* <div *className*="red">{tekst}</div>;

};

export default Second;

I ostatecznie modyfikujemy App.js.

import Second from "./components/Drugi/Drugi";

import First from "./components/Pierwszy/Pierwszy";

import "./App.css";

function App() {

*return* (

    <div>

      <h1>React is great!</h1>

      <p>React is beautiful</p>

      <First *a*={1} *b*={2}></First>

      <Second *tekst*="abc"/>

      <First *a*={4} *b*={2} />

      <Second *tekst*="tekst" />

      <Second *tekst*="test"/>

      <Second *tekst*="Hello world!"/>

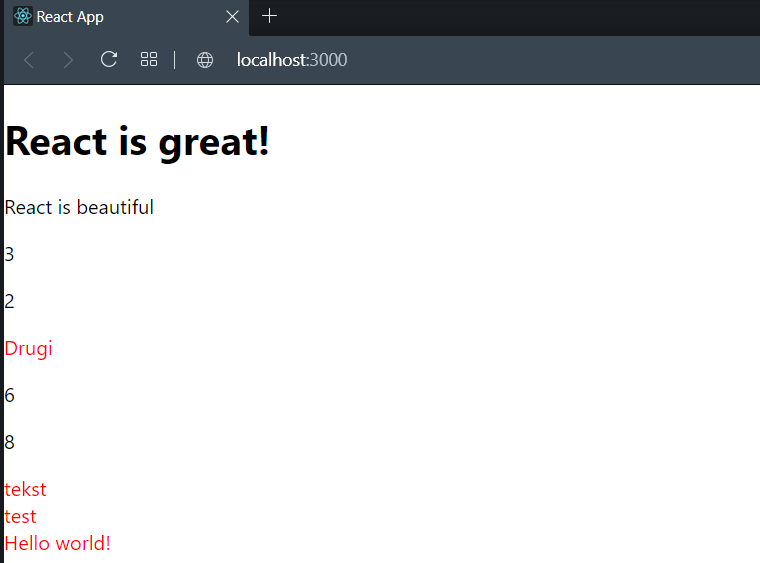
    </div>

  );

}

export default App;

Sprawdźmy wynik w przeglądarce:



Uruchomienie aplikacji z wykorzystaniem propsów

**Zakończenie**

Gratulacje! Ukończyłeś szkolenie wprowadzające do Reacta. Zachęcam Cię do przećwiczenia w praktyce tego wszystkiego, co omówiliśmy oraz ewentualnie przeczytania dokumentacji oraz uzupełnienia wiedzy o dalsze szczegóły.   
  
Poznałeś tutaj podstawowe zagadnienia, trochę teorii oraz stworzyłeś swój pierwszy komponent. W ramach treningu stwórz jeszcze raz projekt React App oraz dodaj do niego kilka komponentów. Zachęcam Cię do samodzielnej eksploracji Reacta.