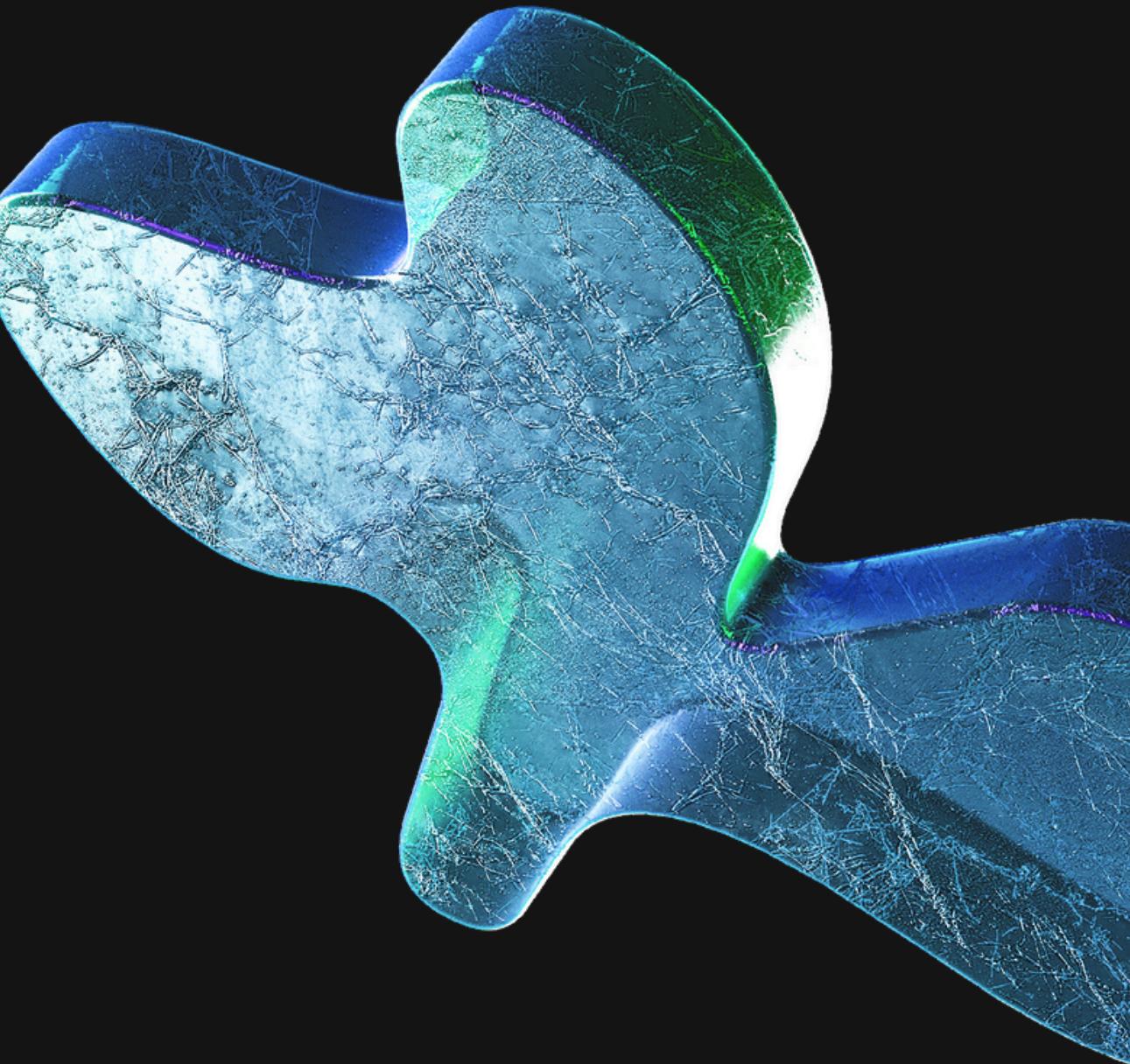
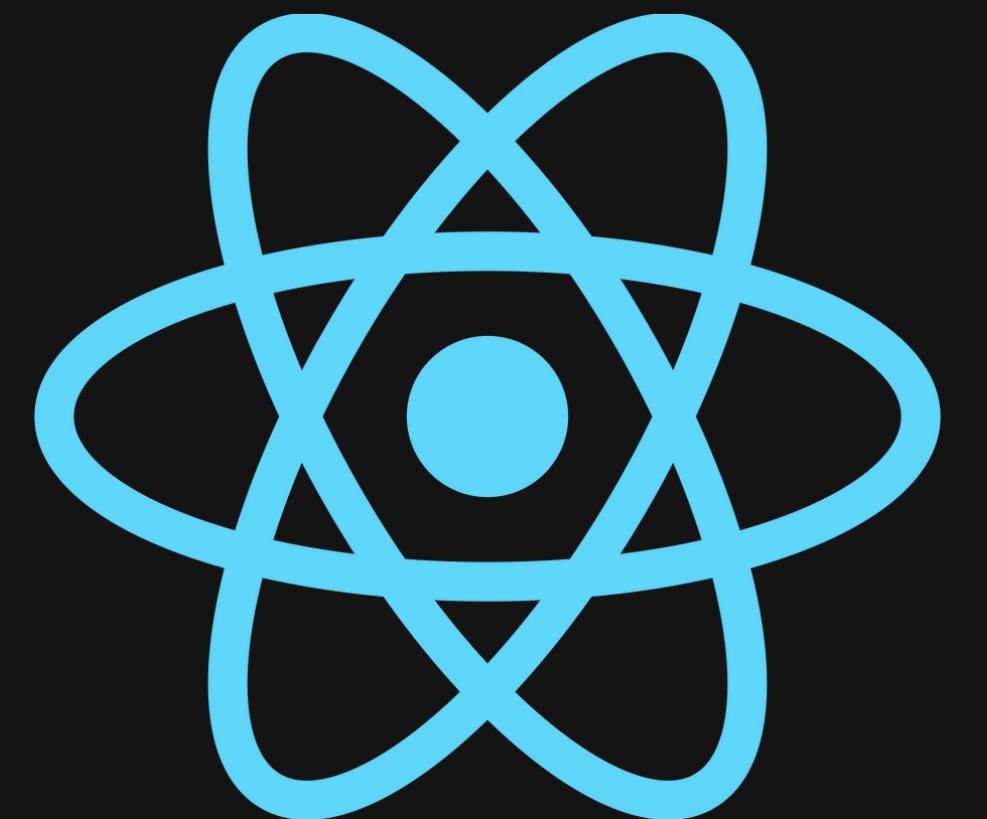


React

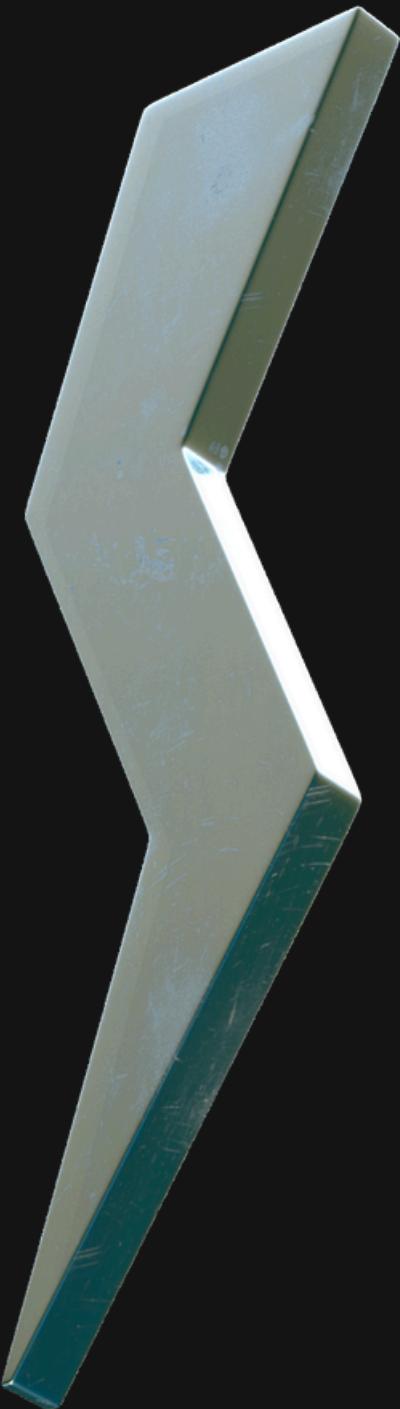


Czym jest react?

Hello, world!

It is 12:26:48 PM.

```
Console Sources Network Timeline
▼<div id="root">
  ▼<div data-reactroot>
    <h1>Hello, world!</h1>
    ▼<h2>
      <!-- react-text: 4 -->
      "It is "
      <!-- /react-text -->
      <!-- react-text: 5 -->
      "12:26:48 PM"
      <!-- /react-text -->
      <!-- react-text: 6 -->
      "."
      <!-- /react-text -->
    </h2>
  </div>
</div>
```



Biblioteka JavaScript

Jego głównym założeniem jest modularność i reużywalność kodu

Aplikacje pisane w oparciu o bibliotekę React są bardzo szybkie

React wykorzystuje wirtualny DOM

React JS jest deklaratywny

Krótka historia

Stworzenie przez Facebooka

Rozwój ekosystemu

Szybka popularność

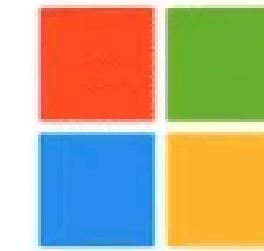
React Native

Wprowadzenie JSX w 2013

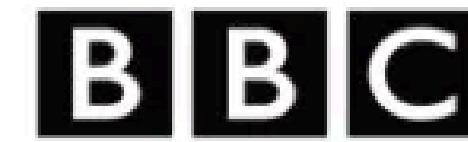
Ciągłe ulepszenia



Companies Using React



Microsoft



NETFLIX



reddit

The New York Times

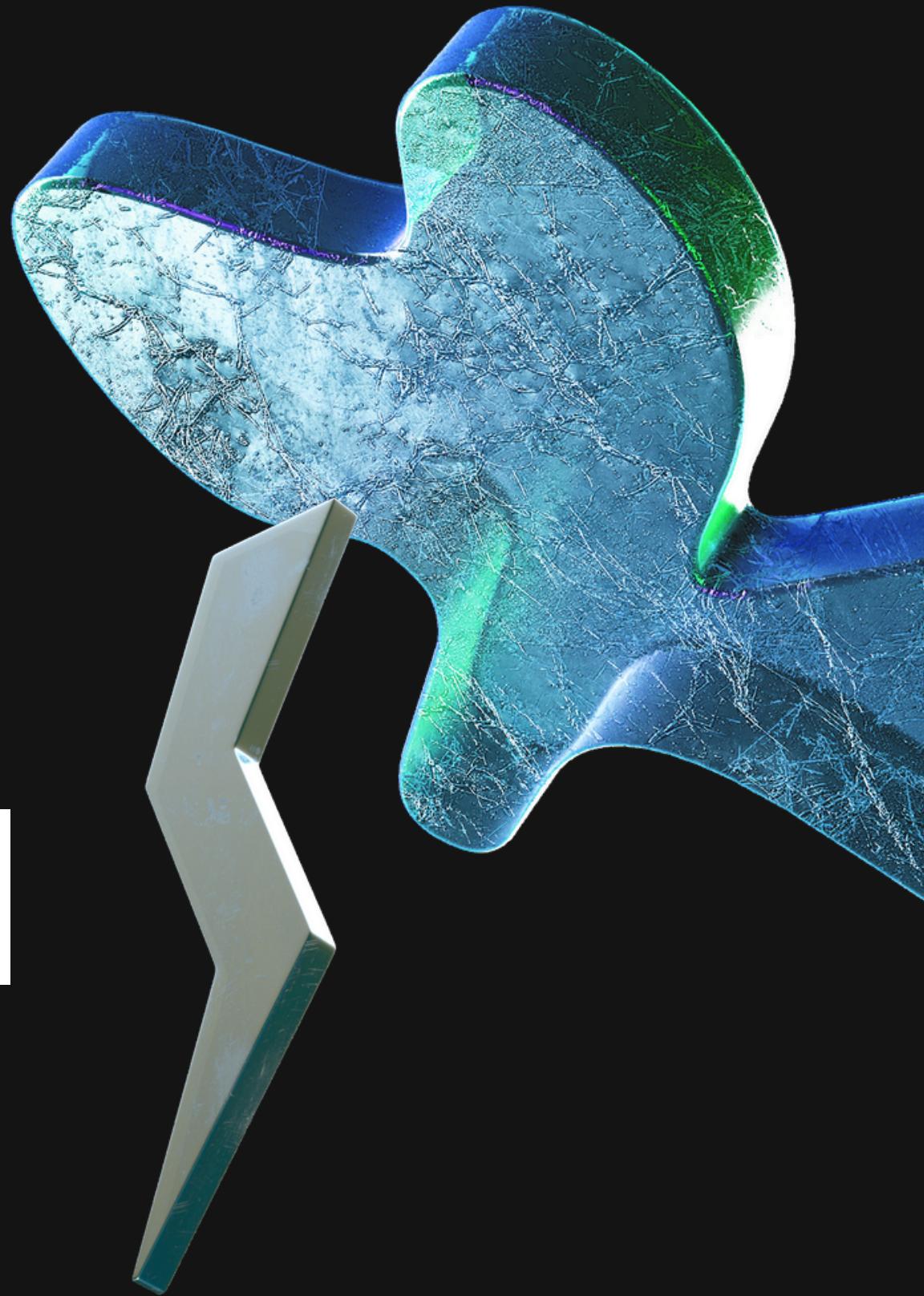


JSX

JSX to skrót od "JavaScript XML"

jest to rozszerzenie które pozwala nam pisać kod HTML i umieszczać je w kodzie React bez konieczności używania takich metod JS jak createElement() czy appendChild(). Dzięki temu możemy łatwo tworzyć dynamiczne strony internetowe lub aplikacje webowe, jak na przykład w React.

```
const myElement = <h1>This is JSX</h1>;
```



XML

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
       more questions later.-->
</quiz>
```

XML



XML

XML jest językiem, czyli służy do przekazywania pewnych treści – NIE jest jednak językiem programowania. Jest językiem znacznikowym, czyli opisuje formę dokumentu, a więc stanowi o sposobie jego interpretacji.

JSX

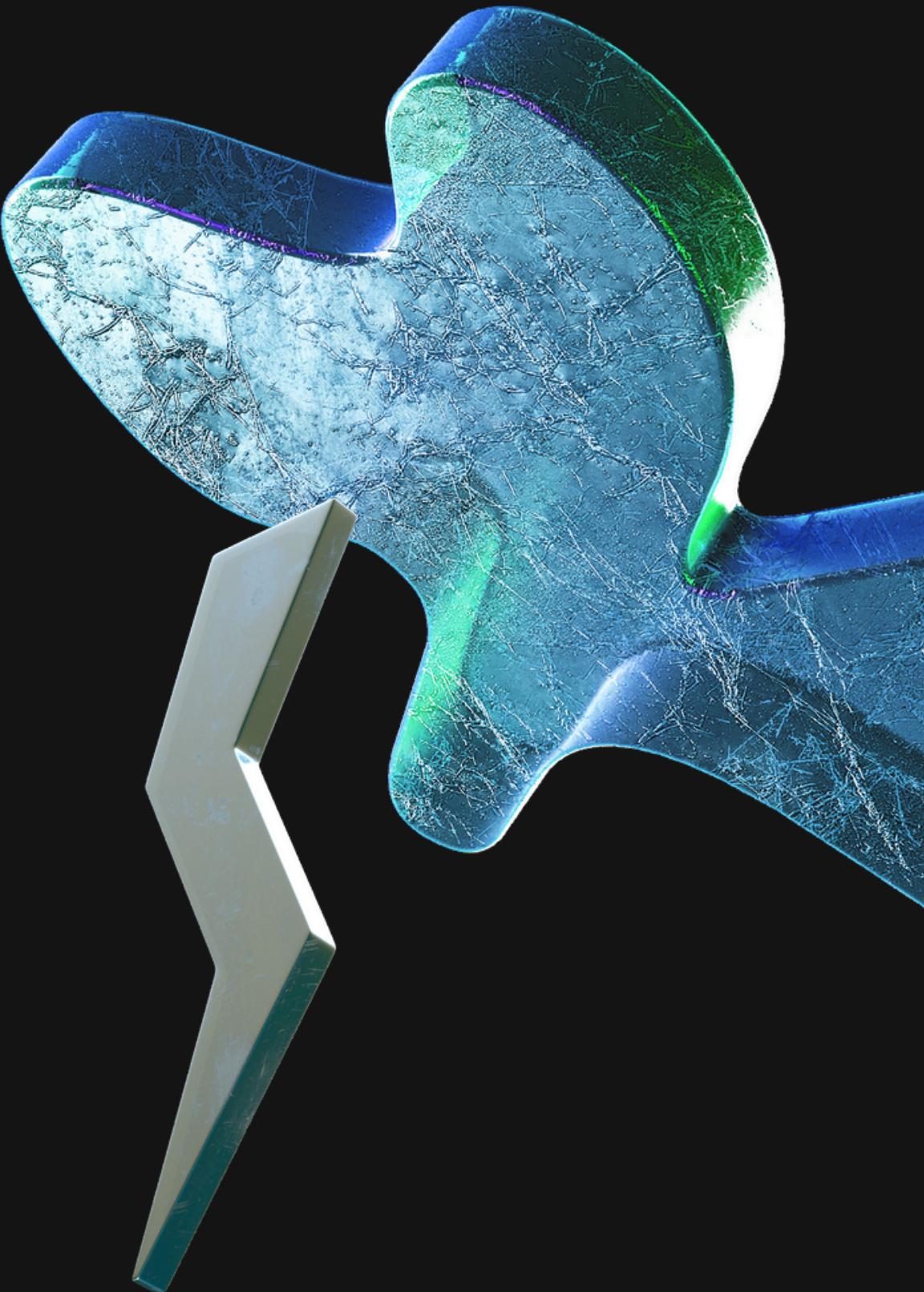
Istnieją 2 sposoby na dodanie elementu HTML w kodzie React:

Pierwszy z nich polega na użyciu metody Reactowej createElement.

```
const myElement = React.createElement( "h1" , { } , "This is JSX" );
```

Drugi, to użycie JSX.

```
const myElement = <h1>This is JSX</h1>;
```



JSX

JSX pozwala również na skonstruowanie bardziej rozbudowanych elementów:

```
const myList = (
  <ul>
    <li> List item </li>
    <li> List item </li>
    <li> List item </li>
  </ul>
);
```

```
const myParagraphs = (
  <p> My text </p>
  <p> My text 2 </p>
);
```

React rzuci błędem

dodanie wrappera

```
const myParagraphs = (
  <div>
    <p> My text </p>
    <p> My text 2 </p>
  </div>
);
```

```
const myParagraphs = (
  <>
    <p> My text </p>
    <p> My text 2 </p>
  </>
);
```

Użycie React Fragments

JSX

Jedną z najważniejszych i najbardziej użytecznych opcji jakie daje nam JSX jest fakt, że możemy do naszych elementów przekazywać dane, mogą to być informację np. ze zmiennych w naszym komponencie. Takie dane zapisujemy w nawiasach klamrowych {}



```
const someData = 5 + 5;  
const myElement = <h1>This is JSX with data { someData }</h1>;
```



Renderowanie

React ma za zadanie renderowanie HTML na stronie internetowej. Używa do tego funkcji createRoot() i jej metody render().



```
<body>
  <div id="root"></div>
</body>
```

index.html

```
const root = ReactDOM.createRoot(
  document.getElementById('root')
);
const element = <h1>Witaj, świecie!</h1>;
root.render(element);
```

klasyczne podejście

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)♦♦
```

użycie Vita do tworzenia struktury aplikacji

Komponenty

wyróżniamy:

- Komponenty funkcyjne

```
function Welcome() {  
  return (  
    <h1>Welcome</h1>  
  )  
  
}  
  
export default Welcome
```

- komponenty klasowe

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Cześć, {this.props.name}</h1>;  
  }  
}
```

To samodzielna i hermetyczna jednostka interfejsu użytkownika, która może zawierać strukturę HTML oraz logikę JavaScript, służąc do wyświetlania danych i zarządzania ich wyglądem i zachowaniem w aplikacji webowej.

Komponenty w React są budulkami, z których składa się interfejs użytkownika, i mogą być łatwo ponownie wykorzystywane w różnych częściach aplikacji.

```
import Welcome from './Welcome'  
  
function App() {  
  return (  
    <>  
    <Welcome/>  
    <Welcome/>  
    </>  
  )  
}  
  
export default App
```

można ich wielokrotnie używać

Props – właściwości

Koncepcyjnie, komponenty są jak javascriptowe funkcje. Przyjmują one arbitralne wartości na wejściu (nazywane “właściwościami” (ang. props)) i zwracają reactowe elementy opisujące, co powinno się pojawić na ekranie.

ZAPIS Z DESTRUKTURYZACJĄ

```
function Welcome({name, surname}) {  
  return (  
    <h1>Welcome {name} {surname}</h1>  
  )  
}
```

ZAPIS Z UŻYCIEM OBIEKTU PROPS

```
function Welcome(props) {  
  return (  
    <h1>Welcome {props.name} {props.surname}</h1>  
  )  
}
```



PRZEKAZYWANIE ARGUMENTÓW

```
function App() {  
  return (  
    <>  
    <Welcome name="Jan" surname="Kowalski" />  
    <Welcome />  
    </>  
  )  
}
```

Props - właściwości

Komponent Reactowy posiada jeden dodatkowy props który możemy użyć, jest nim children i pozwala nam dodać inny element w środku naszego komponentu

UŻYCIE CHILDREN

```
function Welcome(props) {  
  return (  
    <div>  
      <h1>Welcome {props.name} {props.surname}</h1>  
      {props.children}  
    </div>  
  );  
}
```



PRZEKAZYWANIE ARGUMENTÓW

```
function App() {  
  return (  
    <>  
    <Welcome name="Jan" surname="Kowalski" >  
      <p>Some text </p>  
    </Welcome>  
  )  
}
```

Props - właściwości

Określanie jakie typy danych oczekuje się dla właściwości przekazywanych do komponentu. Jest to ważne w celu zapewnienia poprawności danych i uniknięcia błędów.

OKRESLANIE TYPU WŁAŚCIWOŚCI

```
Welcome.propTypes = {  
  name: PropTypes.string.isRequired,  
  surname: PropTypes.string.isRequired,  
  children: PropTypes.node  
};
```

DOMYŚLNE WARTOŚCI

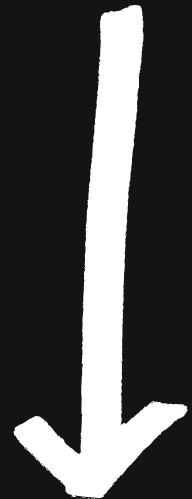
```
Welcome.defaultProps = {  
  name: 'Guest',  
  surname: 'Guest'  
};
```

Props - właściwości

CZĘŚĆ LOGIKI

```
function Component(name, age) {  
  const isAdult = age >= 18;  
  return <p> My name is { name } and I am { isAdult ?  
    "adult" : "not adult" } </p>;  
}
```

ITEROWANIE PO TABLICY



```
function Component({ users }) {  
  return (  
    <div>  
      { users.forEach( (user, index) => <p key={index}>{ user }</p> ) }  
    </div>  
  );  
}
```

Stan komponentu

Co gdybyśmy chcieli klikając w element zmienić jego wartość?

Aby to zrobić musimy dodać kolejny, bardzo kluczowy element jakim jest **state** czyli stan komponentu



Zadeklarowanie zmiennej stanu

W klasie inicjalizujemy stan count z wartością 0, poprzez ustawienie właściwości this.state na { count: 0 } w konstruktorze.

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }
}
```

W komponencie funkcyjnym nie mamy dostępu do this, więc nie możemy przypisywać, ani odczytać wartości właściwości this.state. Zamiast tego wywołamy hook useState bezpośrednio z wewnątrz naszego komponentu:

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

function Example() {
  // Zadeklaruj nową zmienną stanu, którą nazwiemy „count”
  const [count, setCount] = useState(0);
}
```

STATE W KOMPONENTE FUNKCYJNYM

Aby w komponencie funkcyjnym móc korzystać ze state musimy użyć tzw hooka useState.

Spójrzmy zatem jak wygląda dodanie stanu w komponencie funkcyjnym.

- Zaczynamy od zimportowania hooka useState z biblioteki react
- Następnie inicjalizujemy stan wartością domyślną, a useState zwraca 2 wartości: stan, oraz metodę służącą do zmiany wartości stanu

```
import { useState } from "react";

const Component = () => {
    const [ someValue, setSomeValue ] = useState( 0 );
    ...
};
```

STATE W KOMPONENTE FUNKCYJNYM

Mając stan oraz metodę do jego zmiany, możemy użyć ich w naszym komponencie

```
import { useState } from "react";

const Component = ( { name, age } )=> {
    const [ someValue, setSomeValue ] = useState( 0 );

    const handleOnClick = ()=> {
        setState( someValue + 1 );
    }

    return (
        <>
            <p> My name is { this.props.name } and I am { this.props.age } </p>
            <p> Some number value from state: { someValue } </p>
            <button onClick={ handleOnClick }> Kliknij aby zwiększyć wartość o 1 </button>
        </>
    );
};
```

Zalety

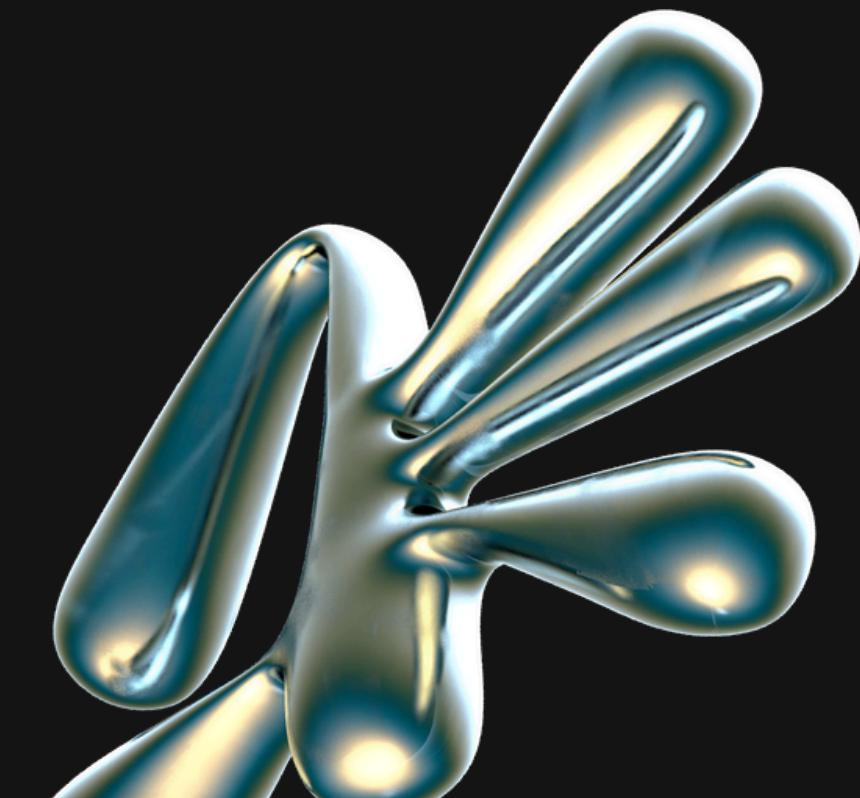


- Niski próg wejścia
- Dojrzałość i stabilność
- Jednokierunkowy przepływ danych
- Bezbolesne SEO
- Open-source
- Nowoczesna aplikacja w pojedynczej komendzie
- Virtual DOM
- Reużywalne komponenty
- React Native

Wady



- Architektura
- JSX
- Zawrotna prędkość rozwoju
- Biblioteki dodatkowe



Pierwszy projekt

Instalacja Node.js i npm: Upewnij się, że masz zainstalowanego Node.js na swoim komputerze. Wraz z Node.js otrzymasz także narzędzie npm (Node Package Manager), które będziemy używać do zarządzania zależnościami projektu.

Download Node.js®

Download Node.js the way you want.

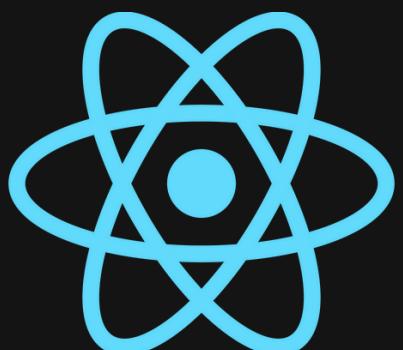
Prebuilt Installer Prebuilt Binaries Package Manager Source Code

I want the **v20.13.1 (LTS)** version of Node.js for **Windows** running **x64**

[Download Node.js v20.13.1](#)

Node.js includes npm (10.5.2) ↗.
Read the changelog for [this version](#) ↗
Read the blog post for [this version](#) ↗
Learn how to [verify signed SHASUMS](#) ↗
Check out all available Node.js [download options](#) ↗
Learn about [Node.js Releases](#) ↗

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\pawel\OneDrive\Pulpit\react> node -v
v20.11.1
PS C:\Users\pawel\OneDrive\Pulpit\react> npm -v
10.2.4
PS C:\Users\pawel\OneDrive\Pulpit\react>
```



Create React App



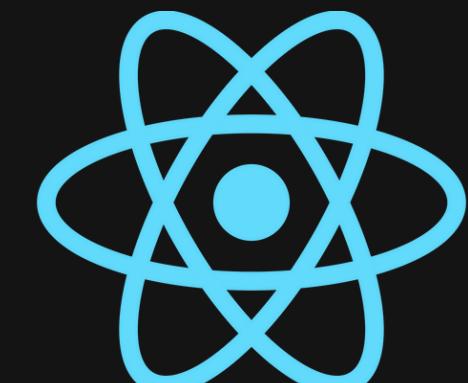
Create React App – narzędzie służące do szybkiej instalacji i konfiguracji projektu w oparciu o bibliotekę React.

Aby skonfigurować projekt przy użyciu CRA należy wykonać poniższą komendą, w której musimy podać nazwę folderu dla naszego projektu

```
npx create-react-app folder_projektu
```

Następnie przechodzimy do utworzonego folderu, uruchamiamy komendę npm start i widzimy, że startuje lokalny serwer z naszą aplikacją.

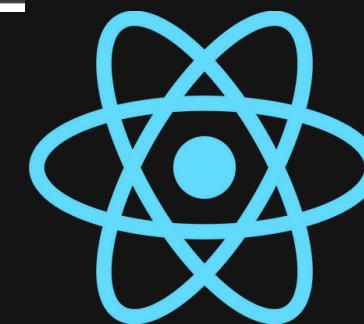
```
cd folder_projektu  
npm start
```



CREATE REACT APP

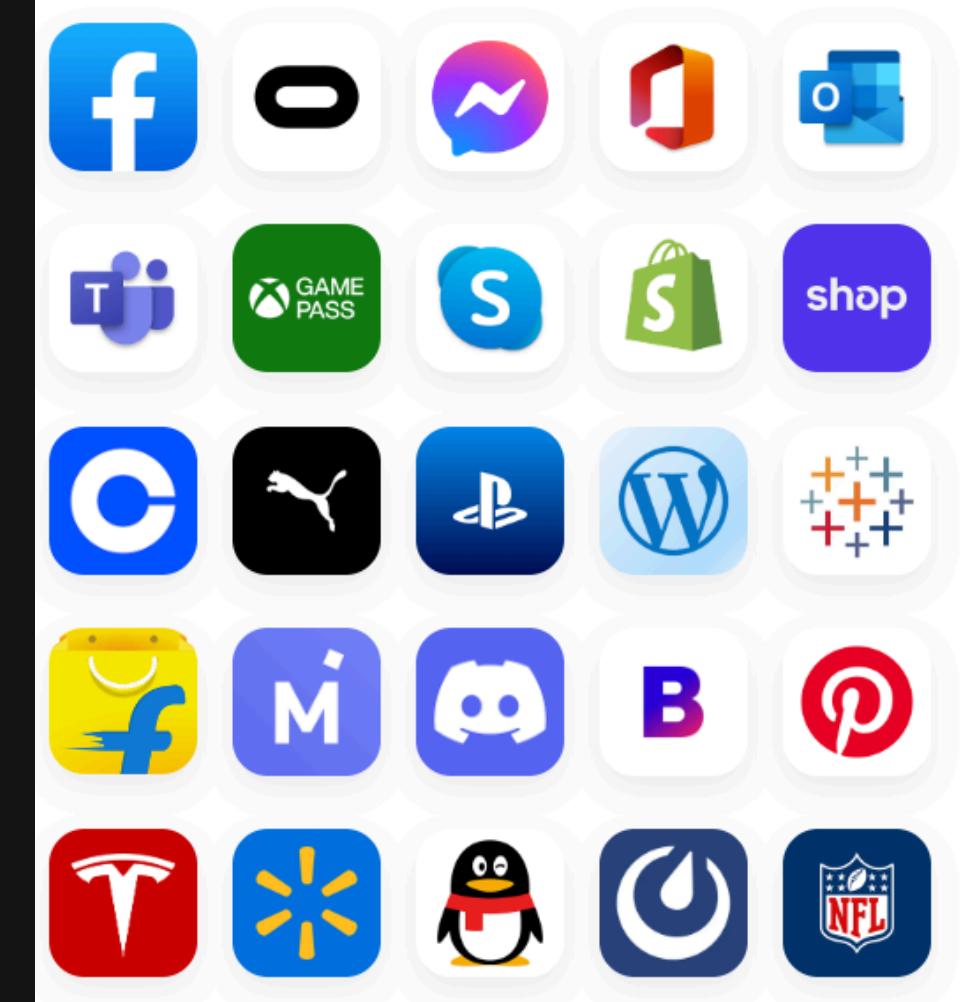
Po uruchomieniu CRA, folder naszego projektu powinien wyglądać mniej więcej tak:

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
└── public
    ├── favicon.ico
    ├── index.html
    ├── logo192.png
    ├── logo512.png
    ├── manifest.json
    └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── serviceWorker.js
    └── setupTests.js
```



React Native

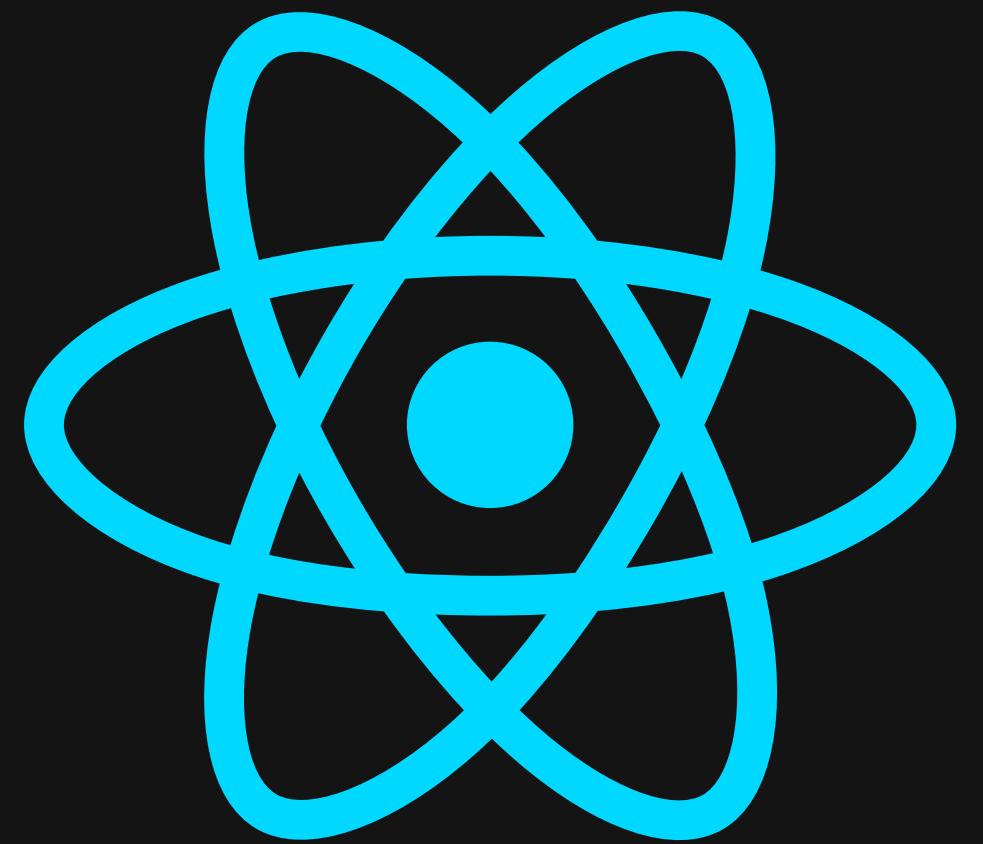
React Native to potężne narzędzie do tworzenia mobilnych aplikacji na platformy iOS i Android, przy użyciu znajomych technologii webowych jak JavaScript oraz biblioteki React. Zamiast pisać oddzielne kody dla każdej platformy, React Native umożliwia tworzenie natywnych aplikacji mobilnych korzystając z jednego, wspólnego kodu źródłowego.



ReactJS i React Native – różnice

- Platforma: React jest do aplikacji webowych, a React Native do aplikacji mobilnych.
- Interfejs użytkownika: React używa elementów HTML, a React Native używa komponentów mobilnych.
- Stylizacja: W React używa się CSS, a w React Native specjalnych właściwości stylu.
- Komponenty specyficzne dla platformy: React nie ma ich, a React Native ma dla iOS i Android.
- Wygląd i zachowanie: React renderuje jako HTML, a React Native jako natywne komponenty mobilne.

Porównanie React z Angular

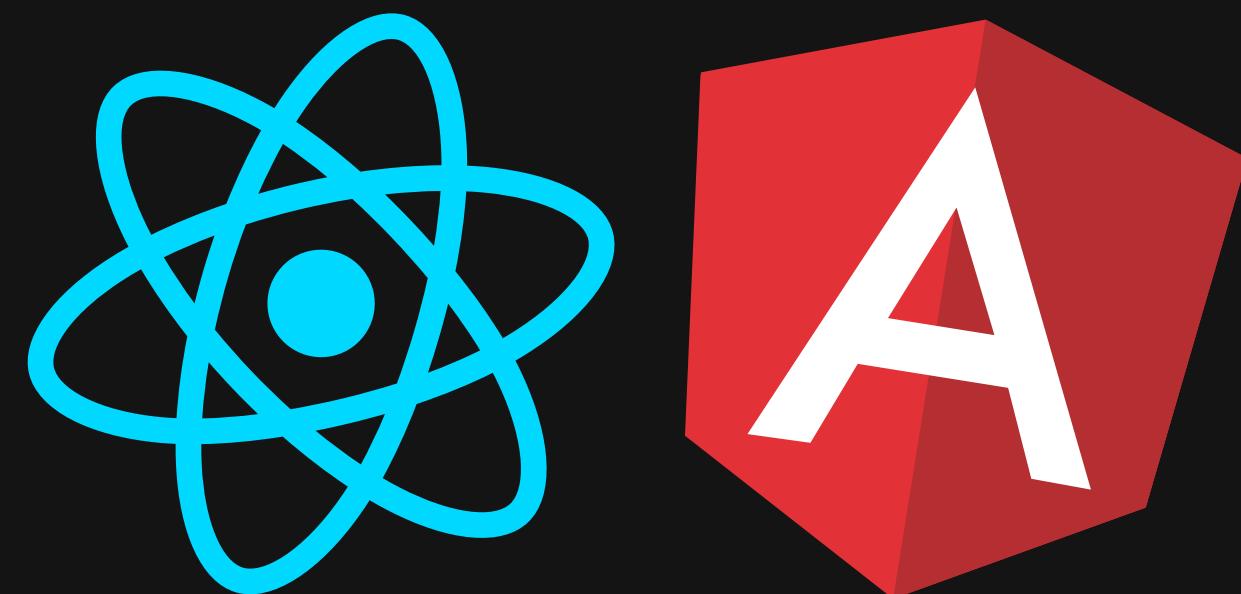


VS

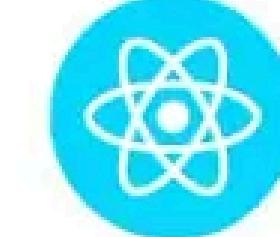


React vs Angular

Angular to framework JavaScriptowy, stworzony przez Google w 2009 roku. Obecnie jest jednym z najstarszych dostępnych frameworków frontendowych. React z kolei to biblioteka frontendowa autorstwa Facebooka, dostępna na rynku od 2013 roku. Od tego czasu zdobyła ogromną popularność i stała się jedną z najczęściej używanych bibliotek w swojej klasie.



- React został utworzony przy użyciu języka JavaScript, podczas gdy Angular jest oparty na TypeScript.
- React jest uważany za łatwiejszy w obsłudze ze względu na dostarczanie podstawowych narzędzi do tworzenia aplikacji, podczas gdy Angular jest uważany za bardziej skomplikowany z powodu bogactwa wbudowanych funkcji i mechanizmów.

ANGULAR JS	VS	REACT JS
		
DEVELOPER		Facebook
TECHNOLOGY TYPE		JavaScript library (View in MVC; requires Flux to implement architecture)
CONCEPT		Brings HTML into JavaScript Works with the virtual DOM Server-side rendering
DATA BINDING		One-way data binding
DEPENDENCIES		Requires additional tools to manage dependencies
LANGUAGE		JavaScript + JSX
Google		
Full-fledged MVC framework written in JavaScript		
Brings JavaScript into HTML Works with the real DOM Client-side rendering		
Two-way data binding		
Manages dependencies automatically		
JavaScript + HTML		

www.creolestudios.com

Koniec

