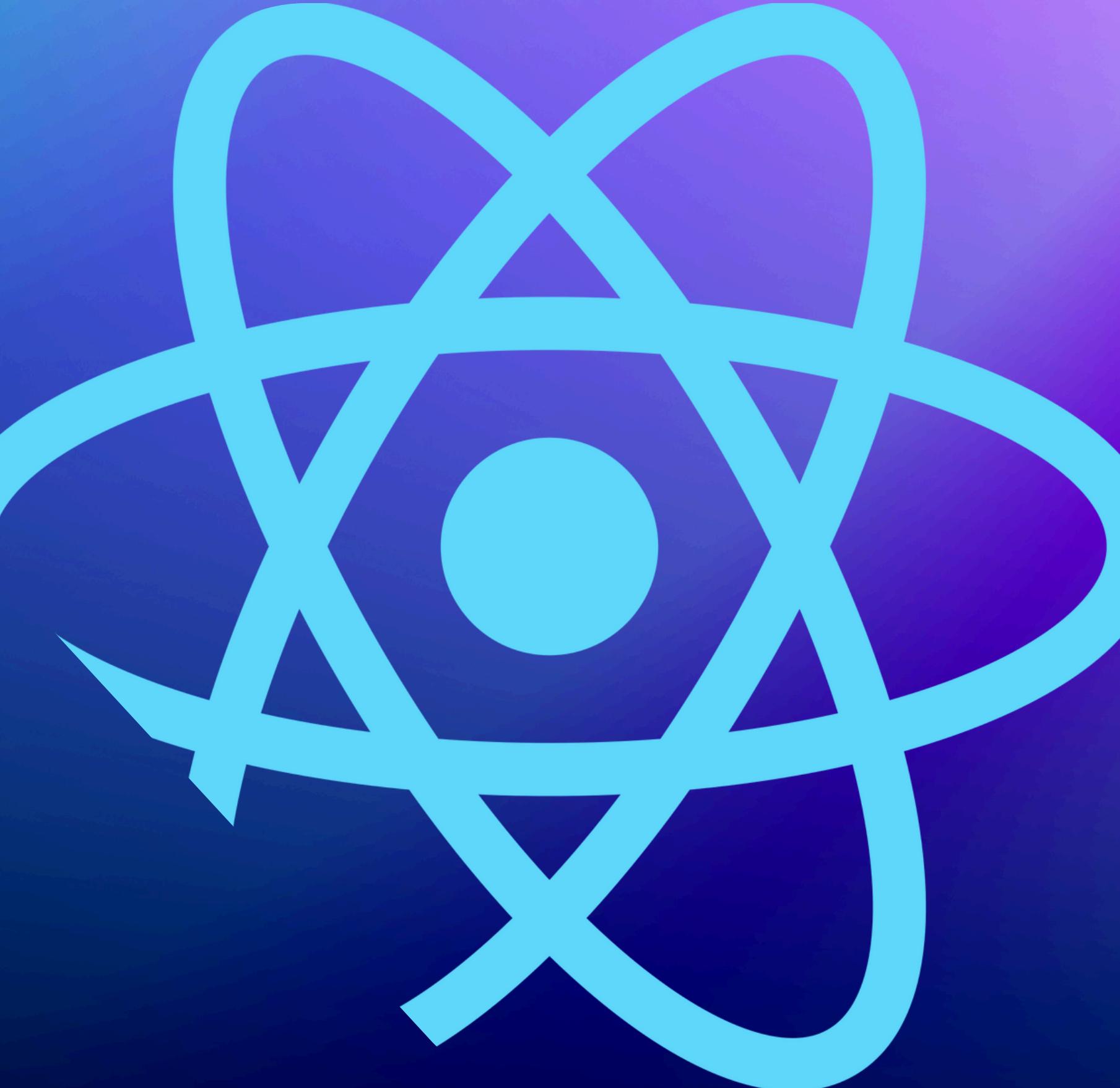


Framework React

*Omówienie zastosowań oraz podstawowych
zasad działania...*



Spis treści

- | | | | |
|---|---|----|--------------------------------------|
| 1 | <i>Czym jest React ?</i> | 7 | <i>ES6 (ECMAScript 6)</i> |
| 2 | <i>Kto stosuje React ?</i> | 8 | <i>Komponenty React</i> |
| 3 | <i>Jak w skrócie działa React ?</i> | 9 | <i>Props, states</i> |
| 4 | <i>Virtual DOM (Document Object Model)</i> | 10 | <i>Renderowanie elementów</i> |
| 5 | <i>Jednokierunkowy przepływ danych</i> | 11 | <i>Zalety oraz wady</i> |
| 6 | <i>XML (Extensible Markup Language) / JSX (JavaScript XML)</i> | 12 | <i>React Native</i> |
| | | 13 | <i>React vs Angular</i> |
| | | 14 | <i>Przykład na żywo</i> |

Czym jest React ?

1

- **biblioteka języka programowania JavaScript, często definiowana jako framework.**
- **dostępna jest na zasadzie open-source**
- **służy do tworzenia interfejsów użytkownika**
- **znajęła szerokie zastosowanie przy budowie aplikacji zgodnie z architekturą SPA (Single-Page Application)**
- **pierwotnie nazwany został FaxJs (2011 rok). Po raz pierwszy zaprezentowany światu pod nazwą React w 2013 roku . Stworzony został przez inżyniera oprogramowania Jordana Walke-a (ówczesny pracownik Facebook)**



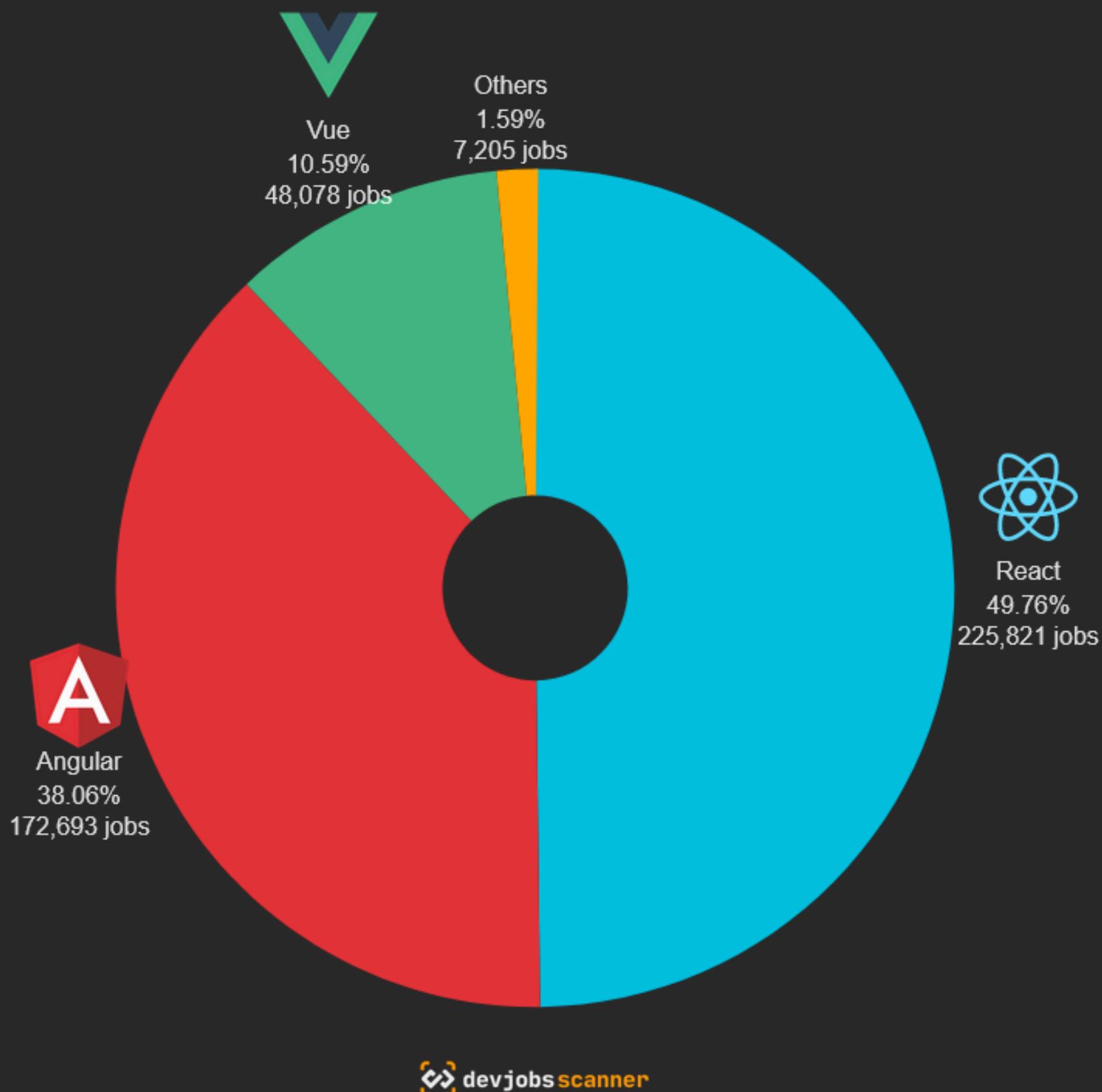
Kto używą React ?

2

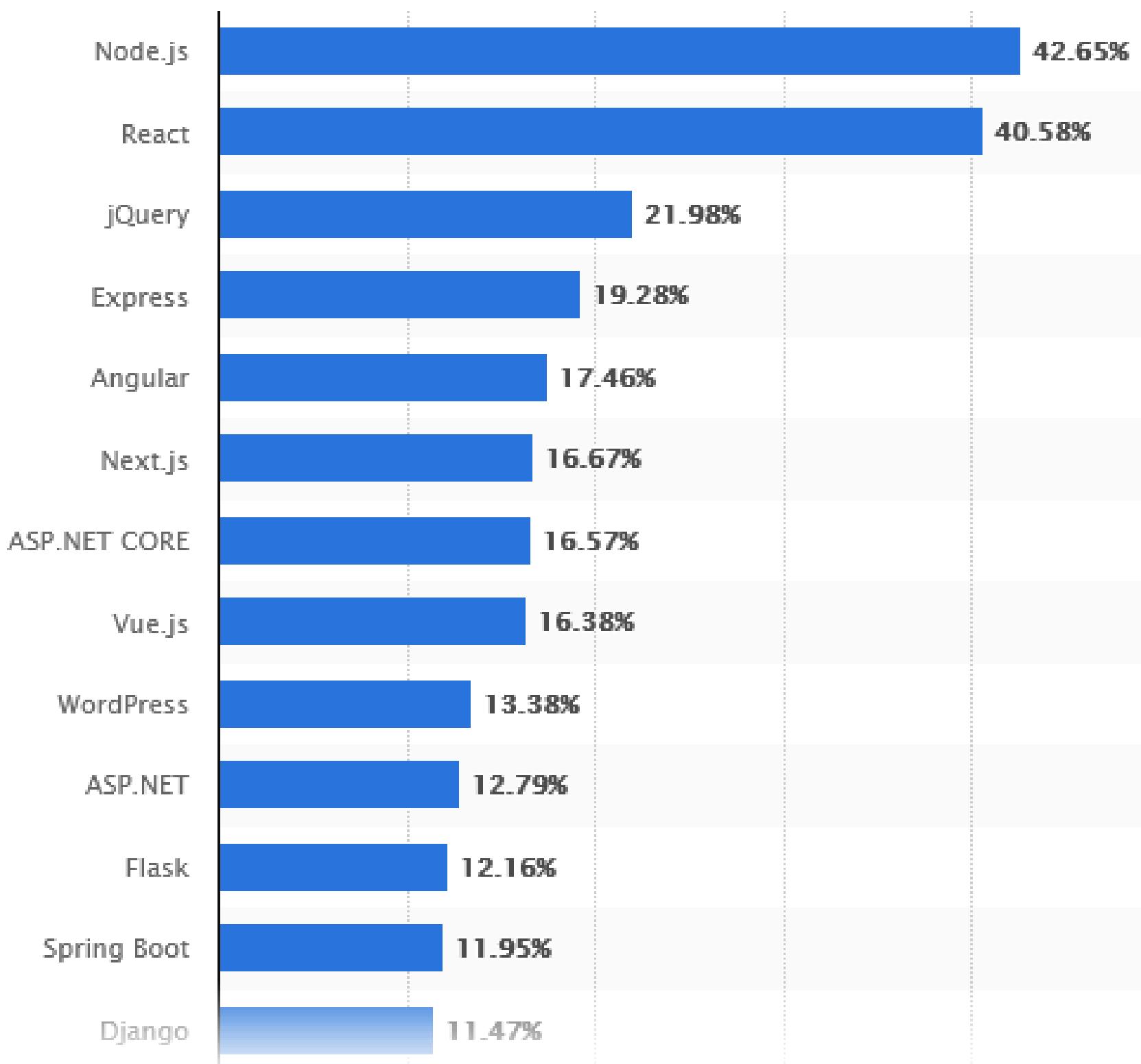


Percentage of jobs by Frontend Framework

From 01-Nov-2022 to 31-Dec-2023



Najczęściej stosowany framework - 2023



źródło: www.statista.com

Dodatkowo React jest jedną z najpopularniejszych bibliotek JavaScript na Github posiadając ponad 222 tysiącami gwiazdek oraz 1600 kontrybutorów

Jak w skrócie działa React ?

3

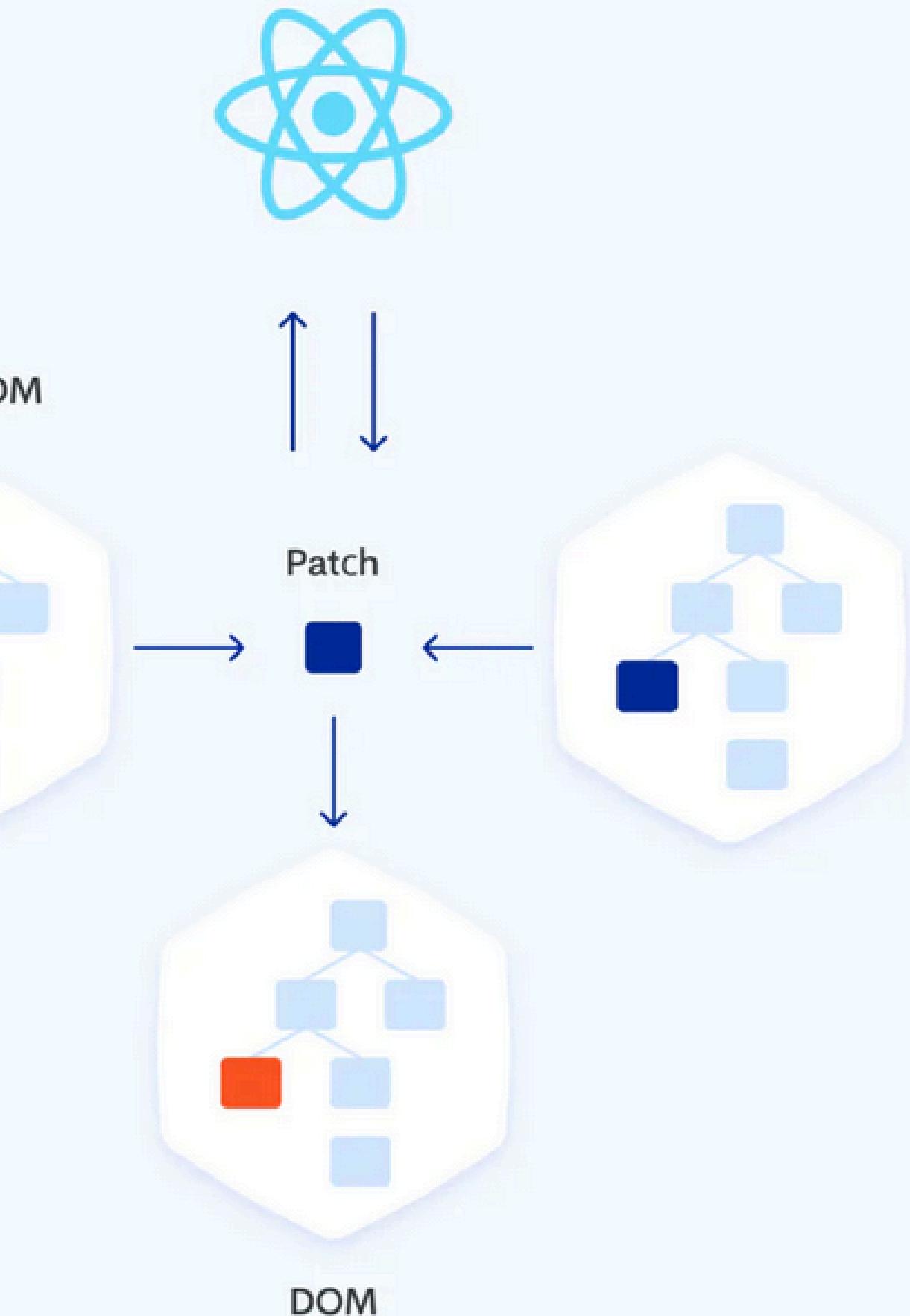
Każdorazowe wpisanie adresu URL w przeglądarce powoduje wysłanie żądania do serwera HTTP. Serwer odsyła następnie wymaganą stronę, która później zostaje renderowana. Taka wymiana występuje w każdej sytuacji, w której chcemy uzyskać dostęp do nowego zasoby np: kolejnej strony.



Co oferuje React ?

React umożliwia nam tworzenie SPA. Zatem dokument HTML w całości jest wczytywany jedynie na początku. Następnie za pomocą JavaScript-u aktualizuje jedynie te elementy, które w danym momencie ulegają zmianom.

A to wszystko dzięki wykorzystaniu Virtual DOM ...



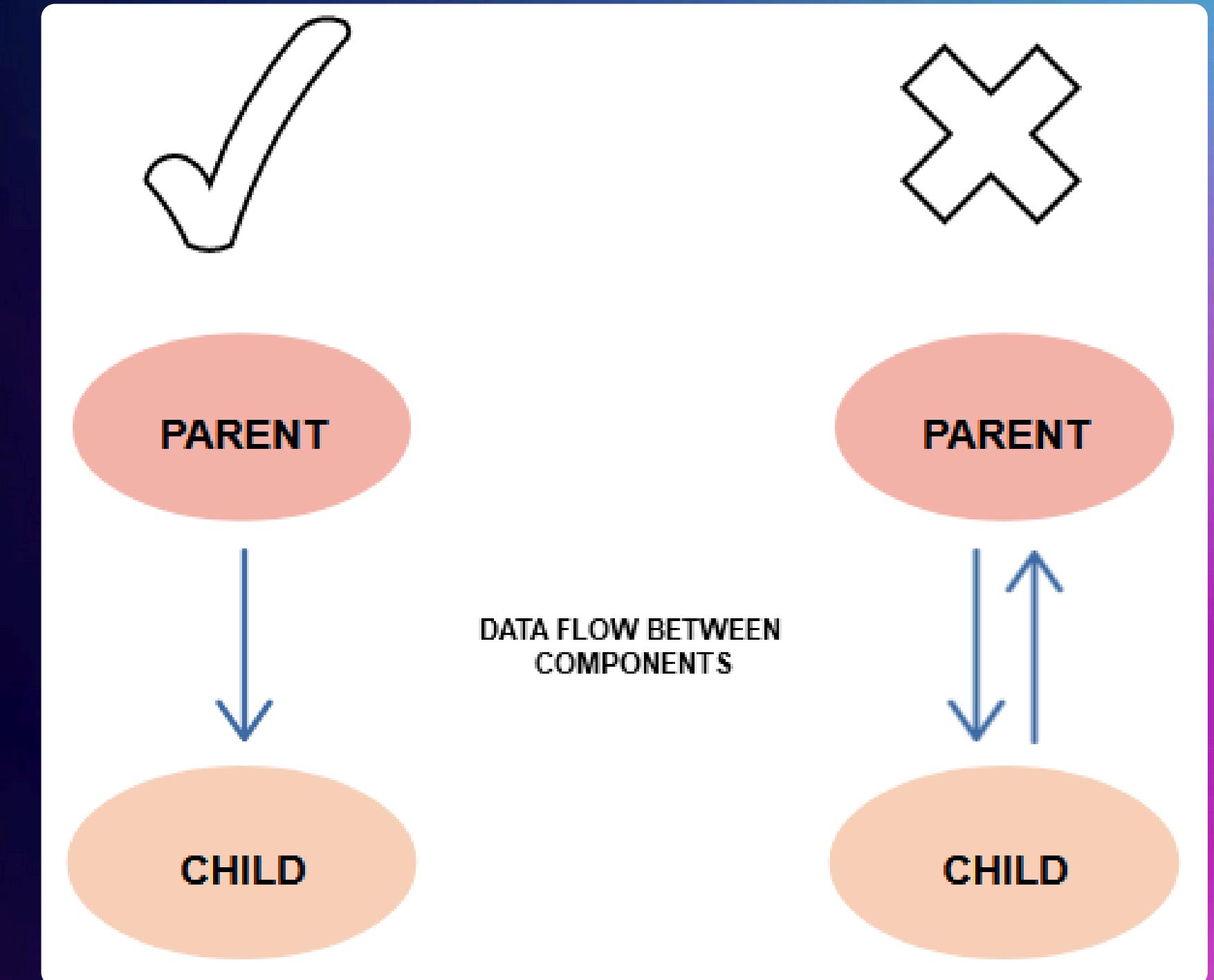
Virtual DOM

Koncepcja Virtual DOM została wprowadzona przez React w celu odwzorowania w pamięci rzeczywistego DOM. Podczas renderowania aplikacji React tworzy wirtualne drzewo, które następnie jest wykorzystywane do określenia minimalnej liczby zmian potrzebnych do zaktualizowania rzeczywistego interfejsu, aby odpowiadał strukturze Virtual DOM.

Jednokierunkowy przepływ danych

5

Przepływ danych w React jest jednokierunkowy.
Oznacza to, że dane płyną jedynie w jednym kierunku - od komponentów “rodziców” w stronę komponentów “dziedzi”. Dane te nazywamy właściwościami(ang. props) i są one zgodnie z konwencją przenaznaczone jedynie do odczytu.



XML to narzędzie pozwalające przechowywać i przesyłać informacje. Definiuje zestaw reguł do kodowania danych, żeby były one łatwo czytelne przez komputer i człowieka.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J. K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

```
<embeddedHTML>
  <title>HTML code embedded in XML</title>
  <description><![CDATA[
    <div>
      <h1>Embedded HTML header</h1>
      <p>Embedded HTML paragraph.</p>
    </div>
  ]]></description>
</embeddedHTML>
```

JSX jest to rozszerzenie składni JavaScript umożliwiające przypisywanie do zmiennych znaczników HTML.

```
function App() {  
  
  const element = <div>Hi</div>  
  
  return element  
}  
  
export default App;
```

osadzanie wyrażeń

```
function App() {  
  const test = "test"  
  const element = <div>{test}</div>  
  
  return element  
}  
  
export default App;
```

wyrażenia warunkowe

```
function App() {  
  let cond = 0;  
  const test = "test"  
  const element = <div>{test}</div>  
  const element1 = <div>Element1</div>  
  
  if (cond == 1) return element  
  else return element1  
}
```



```
function App() {  
  const test = "test"  
  const element = <div className="element">  
    {test}</div>  
  
  return element  
}
```

Jeżeli w obrębie jednego elementu występują elementy potomne należy użyć nawiasów przy definiowaniu zmiennej

```
function App() {  
  const element = (  
    <div>  
      <h1>Tytuł</h1>  
      <h2>Tytuł2</h2>  
    </div>  
  );  
  
  return element;  
}
```

```
return (  
  <>  
    <h1>name: {props.name}</h1>  
    <h1>surname: {props.surname}</h1>  
    <ChildComponent info={infoVariable} />  
  </>  
);  
}
```

JSX jest tłumaczyony jeden do jednego na React-owe elementy (Babel)

```
function App() {  
  const element = (  
    <h1>Witaj</h1>  
  );  
  
  const element1 = React.createElement('h1', null,  
    'Witaj'  
)  
  
  return element;  
}
```

Standardy języka JavaScript

importowania/exportowania modułów

destrukturyzacja

```
import { useState } from "react";
import Alert from "./components/Alert";
import Button from "./components/Button";

function App() {
  const [alertVisible, setAlertVisibility] = useState(false);

  return (
    <div>
      {alertVisible && (
        <Alert onClose={() => setAlertVisibility(false)}>My alert</Alert>
      )}
      <Button onClick={() => setAlertVisibility(true)}>MyButton</Button>
    </div>
  );
}

export default App;
```

arrow func

Komponenty

8

Wszystko jest komponentem w React

Przyjmują one pewne argumenty wejściowe, właściwości (props)

Są to niezależne części kodu

Zwracają elementy HTML

Możliwe jest ich wielokrotne użycie w odrębie aplikacji.

Istnieją dwa rodzaje komponentów: klasowe oraz funkcyjne



```
import './App.css';

const App = (props) => {
  return <div>{props.tekst}</div>
}

export default App;
```

Komponenty funkcyjne

Komponenty klasowe

```
import './App.css'
import React from 'react'

class App extends React.Component {
  render() {
    return <div>{this.props.tekst}</div>
  }
}

export default App
```

Props (właściwości), states (stany)

,

```
index.jsx ×  
1 import ReactDOM from 'react-dom/client';  
2  
3 import { App } from './App.jsx'  
4  
5 ReactDOM.createRoot(  
6   document.querySelector('#root')  
7 ).render(  
8   <App />  
9 )  
10
```

```
App.jsx ×  
1 import React from 'react';  
2  
3 function Welcome(props) {  
4   return <h1>Hello, {props.name}</h1>;  
5 }  
6  
7 export function App(props) {  
8   return (  
9     <div>  
10       <Welcome name="William" />  
11       <Welcome name="Eddie" />  
12     </div>  
13   );  
14 }
```

```
export class Car extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      color: "blue",  
      name: props.name  
    };  
  }  
  
  changeColor = () => {  
    this.setState({ color: "red" })  
  }  
  
  render() {  
    return <div>  
      <h2>I am a {this.state.color} {this.state.name}!</h2>  
      <button type="button" onClick={this.changeColor}>C</button>  
    </div>  
  }  
}
```

podejście klasowe



Przekazywanie danych z elementu nadzawanego do elementu podzewanego (props - właściwości).

```
import React from 'react';

export class Car extends React.Component {
  constructor() {
    super();
    this.state = { color: "blue" };
  }

  changeColor = () => {
    this.setState({ color: "red" })
  }

  render() {
    return <div>
      <h2>I am a {this.state.color} Car!</h2>
      <button type="button" onClick={this.changeColor}>Change Color</button>
    </div>
  }
}
```

podejście klasowe

**dzięki użyciu funkcji
useState() React jest w
stanie śledzić na
bieżąco stan zmiennej
count**



```
index.jsx ×

1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3
4 import { App } from './App.jsx'
5
6 ReactDOM.createRoot(
7   document.querySelector('#root')
8 ).render(
9   <App message="test" color="white"/>
10 )

App.jsx ×

1 import React, { useState } from 'react';
2
3 export function App(props) {
4
5   const [count, setCount] = useState(0);
6
7   const increment = () => {
8     setCount(count + 1);
9   };
10
11 const decrement = () => {
12   setCount(count - 1);
13 };
14
15 return (
16   <div>
17     <p style={{color: props.color}}>{count}</p>
18     <button onClick={increment}>Increment</button>
19     <button onClick={decrement}>Decrement</button>
20   </div>
21 );
22 }
```

Renderowanie elementów (komponentów)

10

*pierwszym krokiem jest wybór tzw
“korzenia”, który będzie
zarządzany przez React DOM*

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3
4 import { App } from './App.jsx'
5
6 const root = ReactDOM.createRoot( document.querySelector('#root') );
7
8 root.render( <App color='white' /> );
```

*następnie do korzenia przekazujemy
komponenty, które chcemy wygenerować
na swojej stronie*

Zalety

- **niski próg wejścia**
- **deklaratywność**
- **wykorzystanie Virtual DOM/Komponentów**
- **renderowanie server-side**
- **JSX**
- **rozwinięte narzędzia deweloperskie (React Developer Tools)**
- **aktywna/rozbudowana społeczność**
- **dowolny backend**

Imperatywne programowanie

```
const root = document.getElementById('root')
const container = document.createElement('section')
const title = document.createElement('h1')
container.id = 'new'
title.innerText = 'Welcome to Our Page!'
container.appendedChild(title)
root.appendChild(container)
```

Deklaratywne programowanie

```
function Title() {
  return (
    <section id="welcome">
      <h1>Welcome to Our Page</h1>
    </section>
  )
}

React.DOM.render(<Title />, document.getElementById("root"))
```

Wady

- ***instensywne tempo rozwoju***
- ***uboga dokumentacja***
- ***niekompletność funkcjonalności***
- ***złożoność integracji z innymi technologiami***

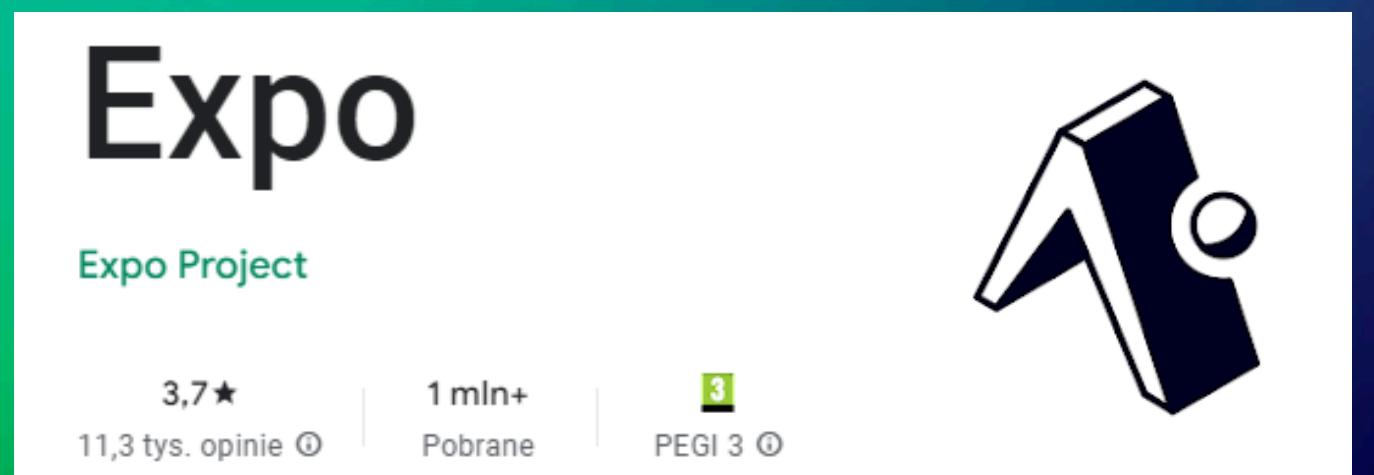
| | |
|---------|-------------------|
| 16.3.1 | 3 April 2018 |
| 16.3.2 | 16 April 2018 |
| 16.4.0 | 24 May 2018 |
| 16.5.0 | 5 September 2018 |
| 16.6.0 | 23 October 2018 |
| 16.7.0 | 20 December 2018 |
| 16.8.0 | 6 February 2019 |
| 16.8.6 | 27 March 2019 |
| 16.9.0 | 9 August 2019 |
| 16.10.0 | 27 September 2019 |

| | |
|---------|-------------------|
| 16.10.1 | 28 September 2019 |
| 16.10.2 | 3 October 2019 |
| 16.11.0 | 22 October 2019 |
| 16.12.0 | 14 November 2019 |
| 16.13.0 | 26 February 2020 |

| | |
|---------|-----------------|
| 16.13.1 | 19 March 2020 |
| 16.14.0 | 14 October 2020 |
| 17.0.0 | 20 October 2020 |
| 17.0.1 | 22 October 2020 |
| 17.0.2 | 22 March 2021 |

Tworzenie aplikacji:

```
npx create-expo-app AwesomeProject  
  
cd AwesomeProject  
npx expo start
```



Składnia jak w "zwykłym" React

```
import React, {useState} from 'react';  
import {Text, TextInput, View} from 'react-native';  
  
const PizzaTranslator = () => {  
  const [text, setText] = useState('');  
  return (  
    <View style={{padding: 10}}>  
      <TextInput  
        style={{height: 40}}  
        placeholder="Type here to translate!"  
        onChangeText={newText => setText(newText)}  
        defaultValue={text}  
      />  
      <Text style={{padding: 10, fontSize: 42}}>  
        {text  
          .split(' ')  
          .map(word => word + '🍕')  
          .join(' ')  
        }  
      </Text>  
    </View>  
  );  
};  
  
export default PizzaTranslator;
```

Możliwość testowania w czasie rzeczywistym na urządzeniu

ANGULAR vs REACT



02. STRUKTURA KOMPONENTÓW

Wykorzystuje Virtual DOM do renderowania komponentów

03. JĘZYK

Opiera się głównie na JavaScript, jednak istnieje możliwość korzystania z TypeScript

04. ROZMIAR I WYDAJNOŚĆ

Znany ze swojej lekkości, ponieważ skupia się jedynie na warstwie widoku architektury MVC

01. ARCHITEKTURA

React to biblioteka JS służąca do tworzenia interfejsów użytkownika

01. ARCHITEKTURA

Angular to kompletny framework oparty o MVC

02. STRUKTURA KOMPONENTÓW

Wykorzystuje hierarchiczną strukturę, wiązanie danych oraz dyrektywy do obsługi DOM

03. JĘZYK

Zbudowany przy pomocy TypeScript i zachęca do jego wykorzystywania w projektach.

04. ROZMIAR I WYDAJNOŚĆ

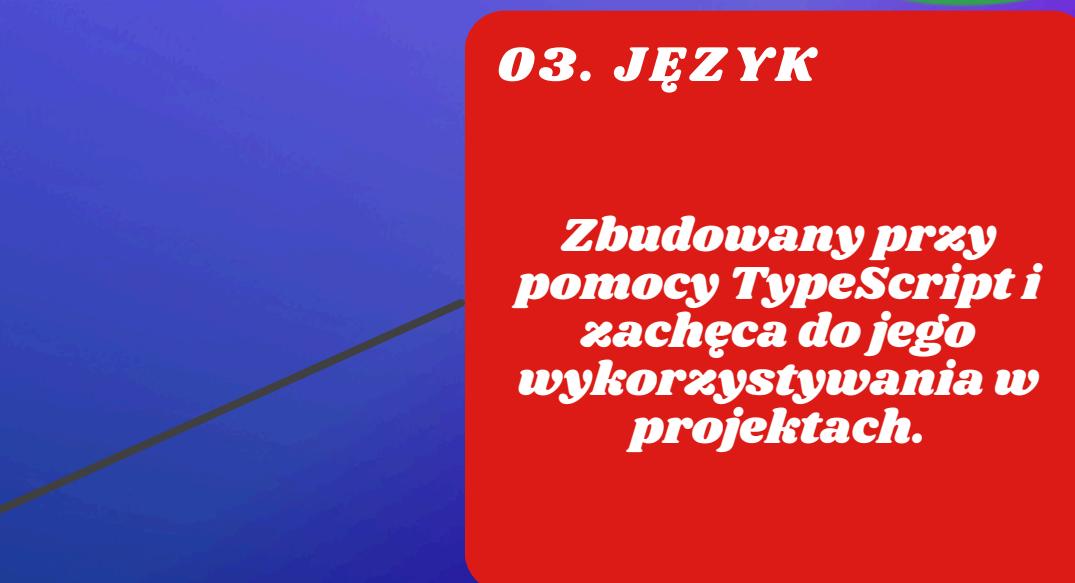
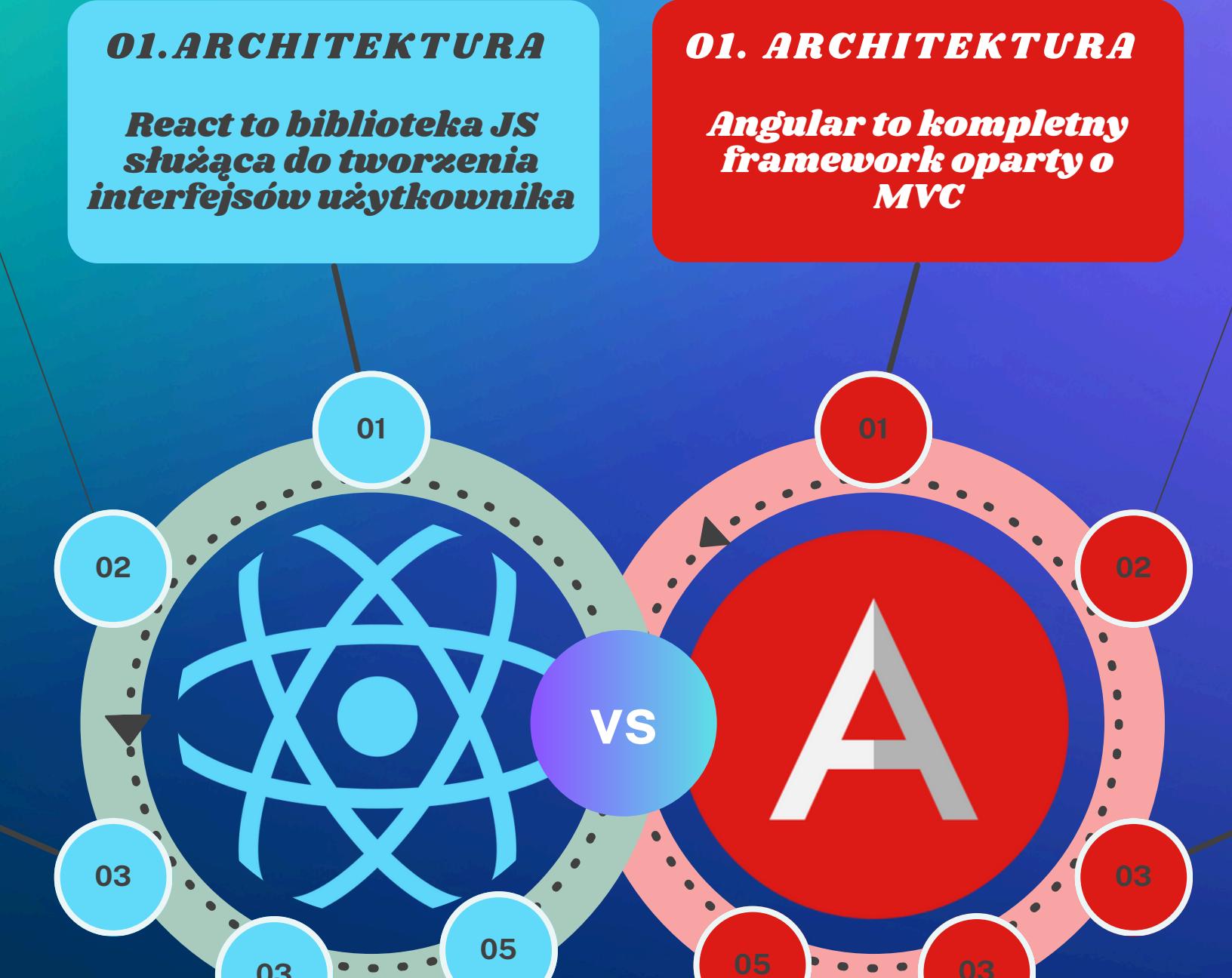
Przez to, że jest framework, może wymagać dodatkowych narzędzi zapewniających wydajność

05. DATA BINDING

Stosuje one-way binding

05. DATA BINDING

Stosuje two-way binding



Jak stworzyć aplikację? - krok po kroku

14

```
PS C:\Users\Jakub\Desktop\Studio\WWW\React> npm create vite@4.1.0
```

- ✓ Project name: ... react_project
- ✓ Select a framework: » React
- ✓ Select a variant: » JavaScript

```
Scaffolding project in C:\Users\Jakub\Desktop\Studio\WWW\React\react_project...
```

```
Done. Now run:
```

```
cd react_project  
npm install  
npm run dev
```



Vite umożliwia nam utworzenie projektu nie tylko w React, a także wybranie wygodnego dla nas języka.



Vite jest to narzędzie do budowania szybkich i lekkich aplikacji. Dzięki wykorzystaniu natywnych modułów zapewnia szybką i płynną pracę

```
PS C:\Users\Jakub\Desktop\Studio\WWW\React> cd ..\react_project\  
PS C:\Users\Jakub\Desktop\Studio\WWW\React\react_project> npm install
```

```
added 74 packages, and audited 75 packages in 6s
```

```
7 packages are looking for funding  
run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
PS C:\Users\Jakub\Desktop\Studia\WWW\React\react_project> npm run dev
```

```
> react_project@0.0.0 dev  
> vite
```

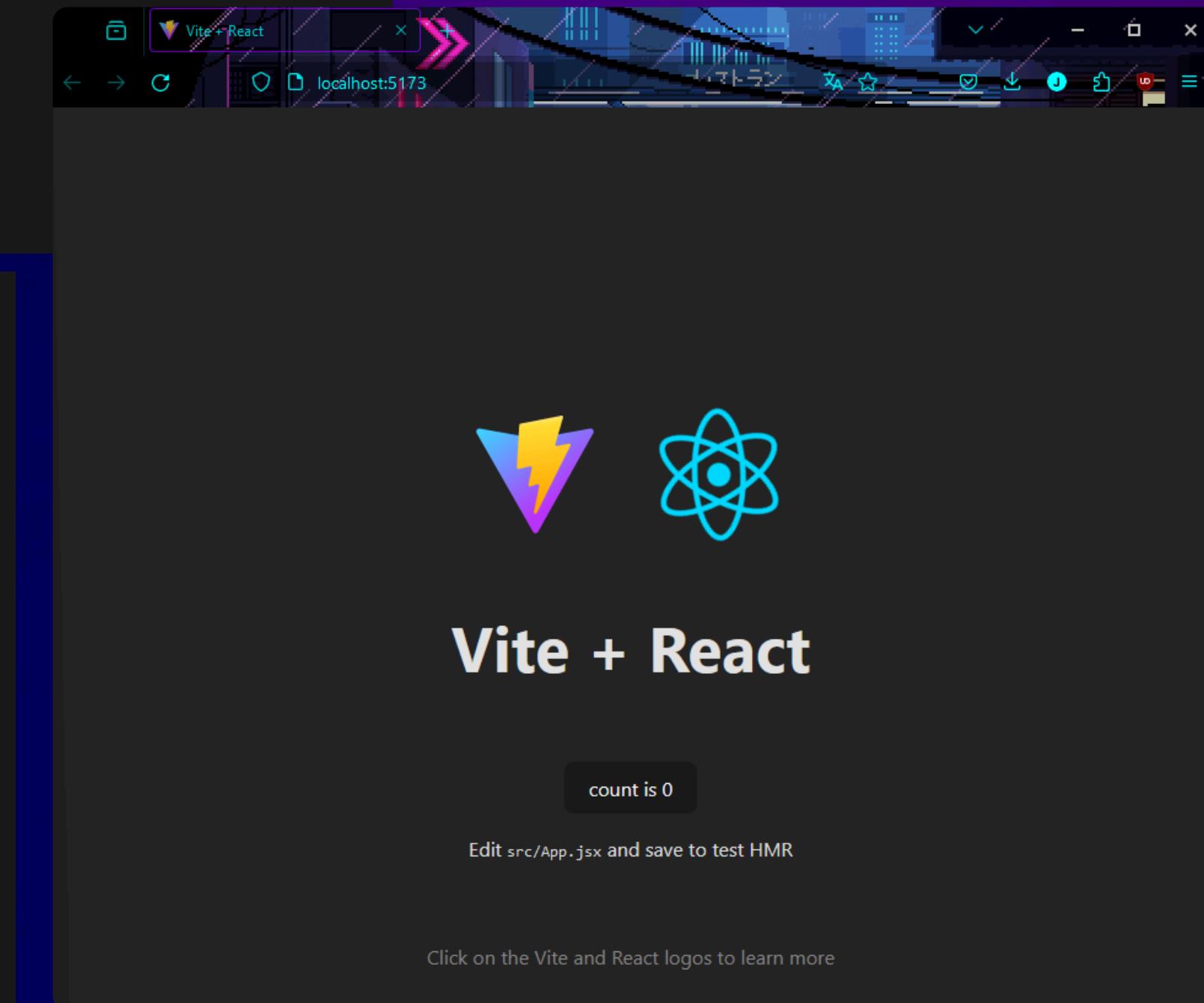
VITE v4.5.3 ready in 437 ms

→ Local: <http://localhost:5173/>
→ Network: use --host to expose
→ press h to show help

zainstalowane biblioteki

```
> node_modules  
> public  
▽ src  
  > assets  
  # App.css  
  ❁ App.jsx  
  # index.css  
  ❁ main.jsx  
  ◆ .gitignore  
  ◁ index.html  
  { } package-lock.json  
  { } package.json  
  JS vite.config.js
```

Za pomocą tej komendy odpalamy serwer na localhost



*Przykład
implementacji i
zastosowania React*

*Dziękujemy za
uwagę*