



# TUTORIAL BIBLIOTEKI REACT

Temat numer 1

# 0 czym prezentacja

---

## Czym jest React?

Krótko i teoretycznie o bibliotece

01



## Programowanie deklaratywne a React

Czym jest i co daje ?

02



## Komponenty Reacta

Jako podstawa budowania aplikacji

03



04

## DOM

Document Object Model



05

## React a HTML i JS XML

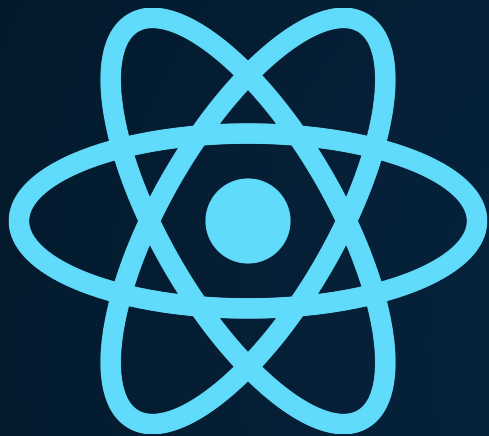
Tworzenie komponentów przy pomocy JavaScript XML



06

## Pierwsza aplikacja

Budowa pierwszej aplikacji i wstępna konfiguracja



**REACT**

# Biblioteka React

---

Darmowa biblioteka JS open-source, którą stworzył programista Facebooka, Jordan Walke w 2013 roku. Nie jest pełnowymiarową biblioteką, wykorzystuje się ją najczęściej do tworzenia prostych interfejsów graficznych w języku programowania JavaScript. Dobra znajomość Reacta pozwala zoptymalizować tworzenie interaktywnych UI.

Obecnie z Reacta korzystają takie korporacje jak: Netflix, Imgur czy PayPal.

Właśnie ze względu na wzrost znaczenia UX/UI w projektowaniu sklepów e-commerce warto poświęcić się nauce Reacta. Obecne trendy stawiają na prosty, minimalistyczny i przejrzysty interfejs, a do tego zdecydowanie nada się React.



# Atuty biblioteki React

---



## ŁATWA

React zapewnia prostą konstrukcję aplikacji, łatwą do zrozumienia strukturę oraz intuicyjne operowanie na obiektach.

Mnogość dokumentacji i poradników biblioteki pozwala na jego szybką oraz efektywną naukę



## WYDAJNA

Struktura bazująca na komponentach i dynamicznym ładowaniu strony w połączeniu z virtual DOM zapewnia wydajną architekturę



## PERSPEKTYWICZNA

React uważany jest aktualnie za bardzo popularne rozwiązanie do tworzenia aplikacji front-endowych. Mnóstwo ofert pracy powoduje, że jest to ciekawa opcja dla każdego, kto myśli o tworzeniu aplikacji



# Programowanie deklaratywne

Biblioteka React wykorzystuje wiele fundamentalnych koncepcji programowania deklaratywnego. Pozwala to na znacznie szybszą implementację konkretnych elementów, a także ich zrozumienie.

# Podejście imperatywne

- `const root = document.getElementById('root');`
- `const paragraph = document.createElement('p');`
- `const paragraphContent = document.createTextNode("I'm a imperative paragraph!");`
- `paragraph.appendChild(paragraphContent);`
- `root.appendChild(paragraph);`

- Przekazujemy krok po kroku CO i JAK ma być wykonane
- Trudniejsze do zrozumienia względem podejścia deklaratywnego

# Podjęście deklaratywne

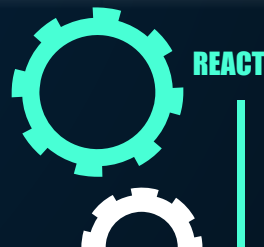
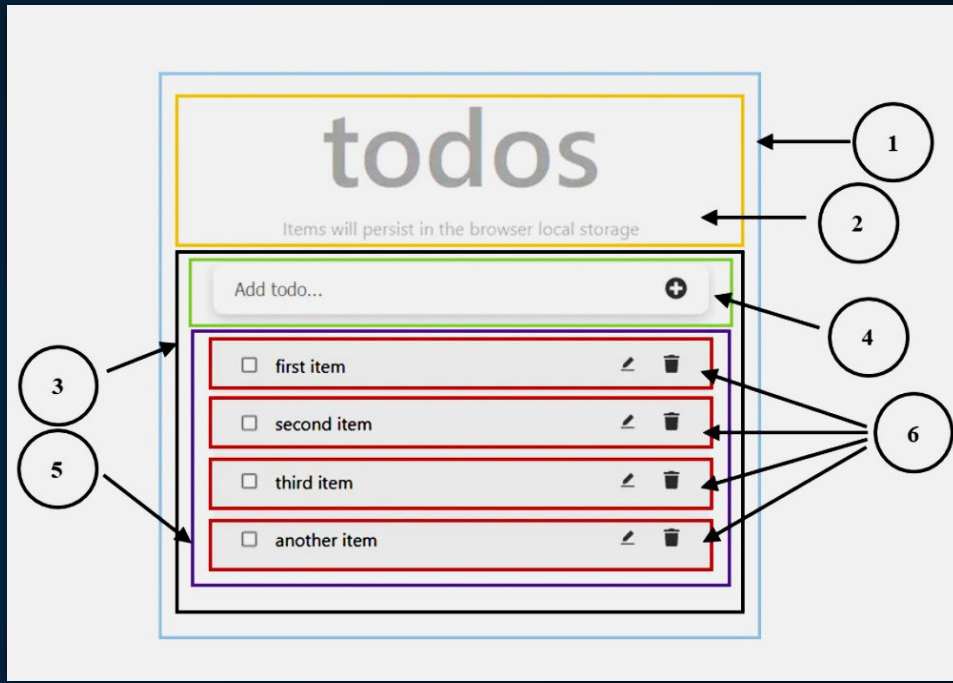
- `<body>`
- `<div id="root"></div>`
- `<p>Im a declarative paragraph!</p>`
- `</body>`

- Przekazujemy tylko CO ma zostać wykonane
- Znacznie szybsza metoda na wykonanie czynności
- Łatwiejsze i przyjemniejsze do zrozumienia
- Szybsze do debugowania

# Komponenty Reacta

Budując jakąkolwiek aplikację, czy to backendową, czy frontendową, warto całą strukturę podzielić na mniejsze elementy o znacznie mniejszym zakresie odpowiedzialności.

Posługując się Reactem, tworząc aplikację dzielimy interfejs na poszczególne elementy nazywając je komponentami, które pozwalają nam na podzielenie logiki systemu w wygodny i przejrzysty sposób.





# Atuty podziału na komponenty



Większa kontrola nad kodem



Kod łatwiejszy do testowania



Przejrzystszy kod



Kod wygodny do edycji



Separacja odpowiedzialności





# DOM (Document Object Model)

Obiektowy model dokumentu reprezentujący w pamięci RAM komputera całą strukturę dokumentu HTML wygenerowaną dzięki przesłanym tekstowo informacjom (front-endowym zasobom witryny).

DOM zapewnia także językowi JavaScript sposoby modyfikacji istniejącej w pamięci RAM struktury z poziomu skryptu.

Hierarchia DOM w połączeniu z językiem JS stanowi dla programisty webowego tzw. API - interfejs programowania aplikacji webowej.



# DOM ciąg dalszy

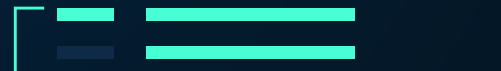
DOM możemy modyfikować poprzez tworzenie, modyfikowanie i dołączanie węzłów oraz atrybutów.

## Metody dokumentu

- `document.getElementById()` – zwraca element o atrybucie `id`
- `document.getElementsByTagName(znacznik)` – zwraca tablica zawierająca elementy o podanej nazwie znacznika `symbol`  
wieloznacznik \* pozwala otrzymać tablicę wszystkich węzłów dokumentu
- `document.createTextNode(tekst)` – tworzy nowy węzeł tekstowy zawierający podany tekst
- `document.createElement(znacznik)` – tworzy nowy element html, po utworzeniu można go dodać do dokumentu
- `document.documentElement` – obiekt reprezentujący sam dokument, pozwala na odczyt informacji o dokumencie

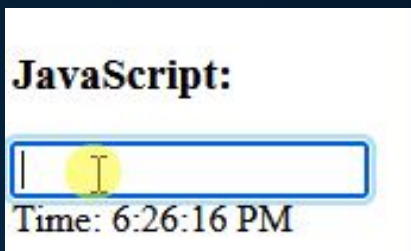
## Metody węzłów

- `appendChild(nowy)` – dołącza wskazany nowy węzeł po wszystkich istniejących węzłach obiektu
- `insertBefore(nowy, istniejący)` – wstawia nowy węzeł potomny przed wskazanymi istniejącymi węzłami potomnymi
- `replaceChild(nowy, stary)` – zastępuje wskazany węzeł nowym
- `removeChild(węzeł)` – usuwa wskazany węzeł
- `cloneNode()` – tworzy kopię istniejącego węzła
- `getAttribute(nazwa atr)` – pobiera wartość wskazanego atrybutu, zapisuje w zmiennej
- `setAttribute(nazwa atr)` – ustawia wartość atrybutu
- `removeAttribute(nazwa atr)` – usuwa dany atrybut



# Przykład renderu strony przy pomocy DOM


- `const update = () => {`
- `// JavaScript`
- `const element2 = ``
- `<h3>JavaScript:</h3>`
- `<form>`
- `<input type="text"/>`
- `</form>`
- `<span>Time: ${new Date().toLocaleTimeString()}</span>`
- ``;`
- `document.querySelector('#root2').innerHTML = element2;`
- `};`
- `setInterval(update, 1000);`



```
<!--JavaScript container-->
<div id="root2">
  <h3>JavaScript:</h3>
  <form> <input type="text"/> </form>
  <span>Time: 6:26:15 PM</span>
</div>
```



## VIRTUAL DOM

- Dostarczany przez Reacta
  - Wirtualna reprezentacja DOM
  - Przy renderze strony Wirtualny DOM jest przypisywany do pamięci i przy każdej zmianie UI React tworzy nowego wirtualnego DOMa po czym porównuje go do poprzedniego stanu co pozwala na aktualizację tylko konkretnych węzłów i dużo lepszą wydajność względem standardowego DOMa, który renderuje na nowo wszystkie węzły
- 

# Przykład renderu strony przy pomocy Virtual DOM

- `<body>`
- `<h1>React in HTML</h1>`
- `<!-- Kontener Reacta -->`
- `<div id="root1"></div>`
- `<!-- Skrypt Reacta -->`
- `<script`
- `src="https://unpkg.com/react@18/umd/react.development.js"`
- `crossorigin`
- `></script>`
- `<script`
- `src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"`
- `crossorigin`
- `></script>`
- `<!-- Załadowanie skryptu -->`
- `<script src="script.js"></script>`
- `</body>`



```
<!--React container-->
▼ <div id="root1"> [event]
  ▼ <div>
    ▼ <form>
      <input type="text"> [event]
    </form>
  ▼ <span>
    Time:
    8:13:17 PM
  </span>
</div>
```

# Przykład renderu strony przy pomocy Virtual DOM

```
• const rootElement = document.querySelector("#root1");
• const root = ReactDOM.createRoot(rootElement);
• const update = () => {
•   const element1 = React.createElement(
•     'div',
•     null,
•     React.createElement(
•       'form',
•       null,
•       React.createElement('input', {
•         type: 'text',
•       })
•     ),
•     React.createElement(
•       'span',
•       null,
•       'Time: ',
•       new Date().toLocaleTimeString()
•     )
•   );
•   root.render(element1);
• };
• setInterval(update, 1000);
```

## React in HTML

Time: 8:13:17 PM

```
<!--React container-->
▼ <div id="root1"> [event]
  ▼ <div>
    ▼ <form>
      <input type="text"> [event]
    </form>
    ▼ <span>
      Time:
      8:13:17 PM
    </span>
  </div>
```



# JSX

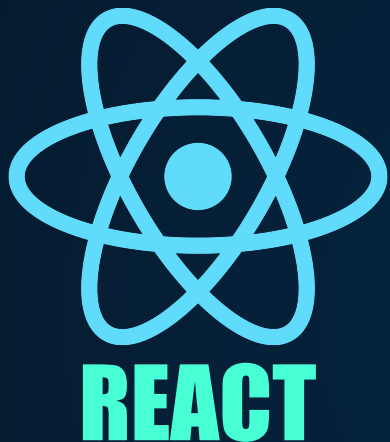
- Rozszerzenie do JS
- Inaczej JavaScript XML
- Pozwala na pisanie w języku HTML w React
- Ułatwia użytkownikowi definiowanie UI

Przykład :

- ```
let element1 = (  
  •   <div>  
  •     <form>  
  •       <input type="text" />  
  •     </form>  
  •     <span>Time: {new Date().toLocaleTimeString()}</span>  
  •   </div>  
  • );
```







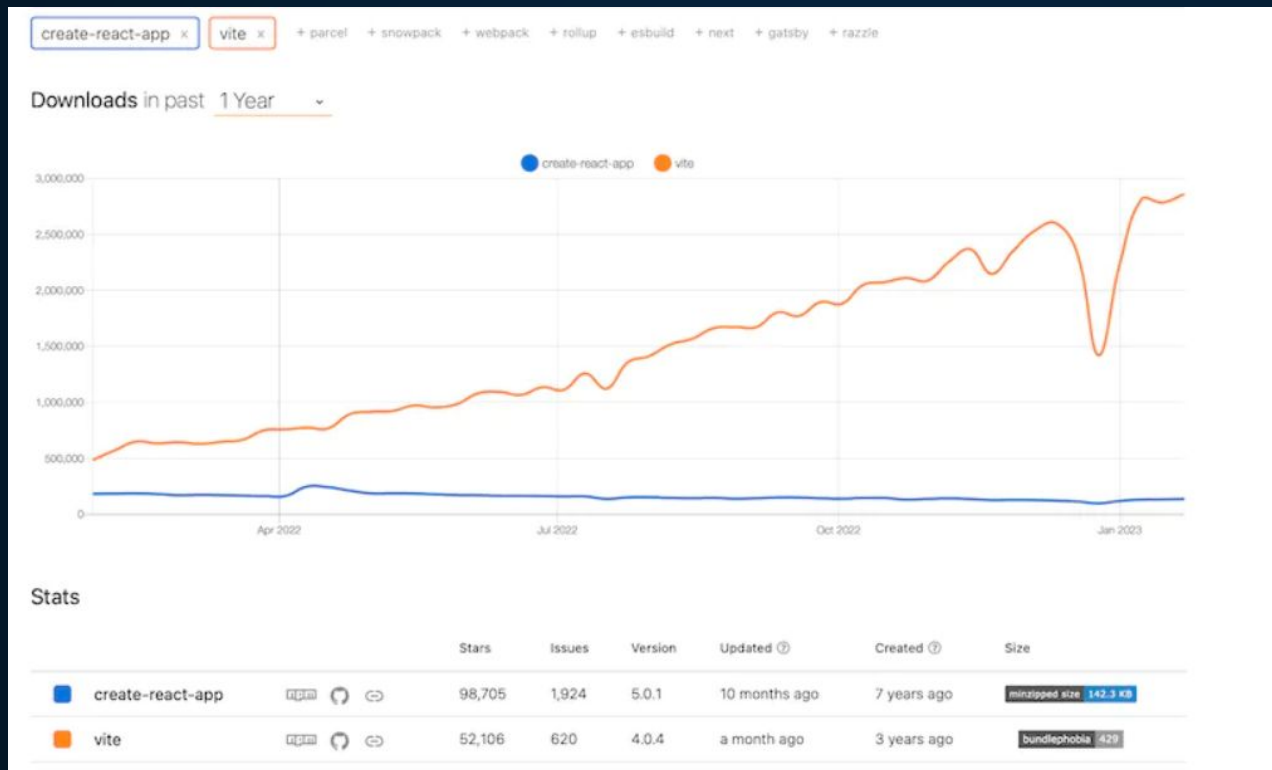
# Przystosowanie środowiska do pracy

---

- Utworzenie projektu przy pomocy CLI (niezbędny node.js i npm)
- Uruchomienie projektu (npm start) na porcie 3000
- Utworzenie projektu przy pomocy Vite



# CLI vs Vite



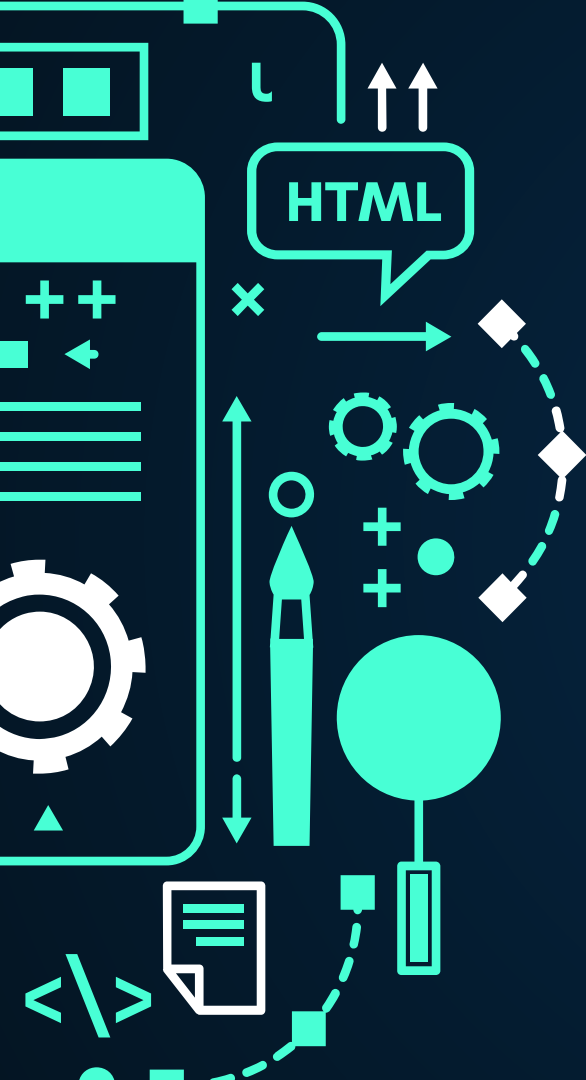
## Ostatni Rok dla Vite

**+70%**

Wzrost popularności co miesiąc

**+1450%**

Większa popularność względem CLI na początku 2023



# Koniec Tematu Numer 1

Przydatne linki :

<https://nodejs.org/en/download/> - Node.js do pobrania

<https://reactjs.org/docs/getting-started.html/> - dokumentacja Reacta

<https://vitejs.dev/guide/> - dokumentacja Vite

Źródło wiedzy :

<https://ibaslogic.com/react-tutorial-for-beginners/>

