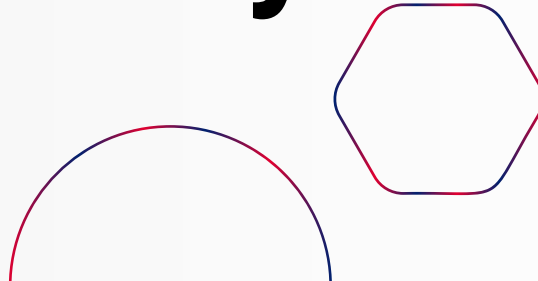




# **Komponenty REACTa i Modele Danych**

Temat numer 2



# Budowa Aplikacji

Aplikacje React są budowane poprzez łączenie fragmentów zaizolowanych komponentów. Te komponenty są fragmentami kodu wielokrotnego użytku i mogą być oparte na funkcjach lub klasach.



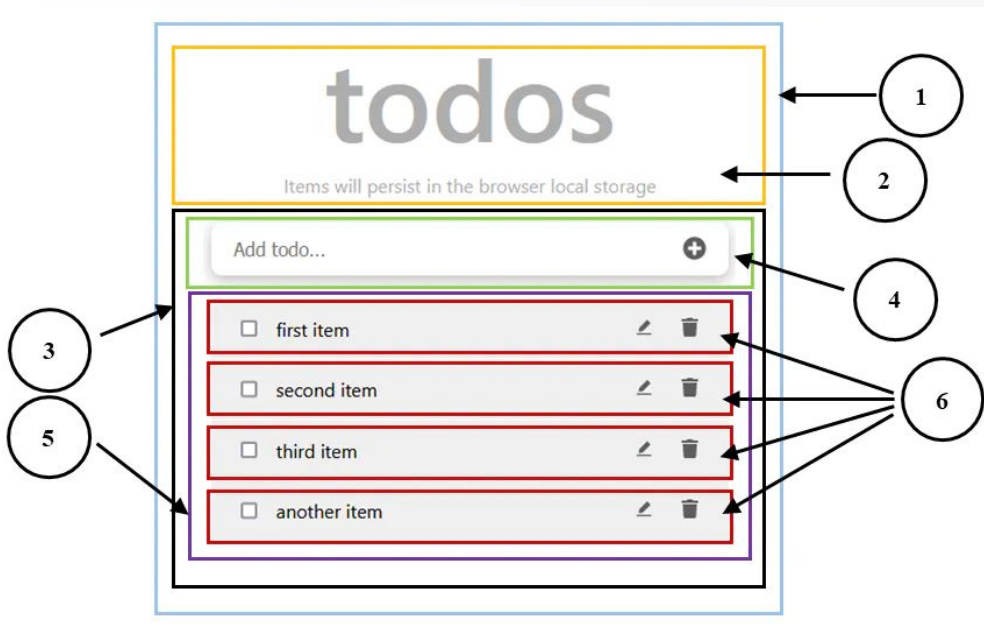
# Komponenty Klas i Funkcji REACT



Komponent klasy React to komponent, który tworzy się przy użyciu składni klasy ES6, podczas gdy komponent funkcyjny tworzy się, pisząc funkcję. Komponent funkcyjny opisuje zachowanie aplikacji, w tym renderowanie elementu JSX.

Jak każdy element, stan elementów JSX wewnątrz komponentu również może ulec zmianie. React może skutecznie „reagować” na zmiany stanu, aby zachować aktualność modelu DOM. To sprawia, że React jest biblioteką reaktywną.

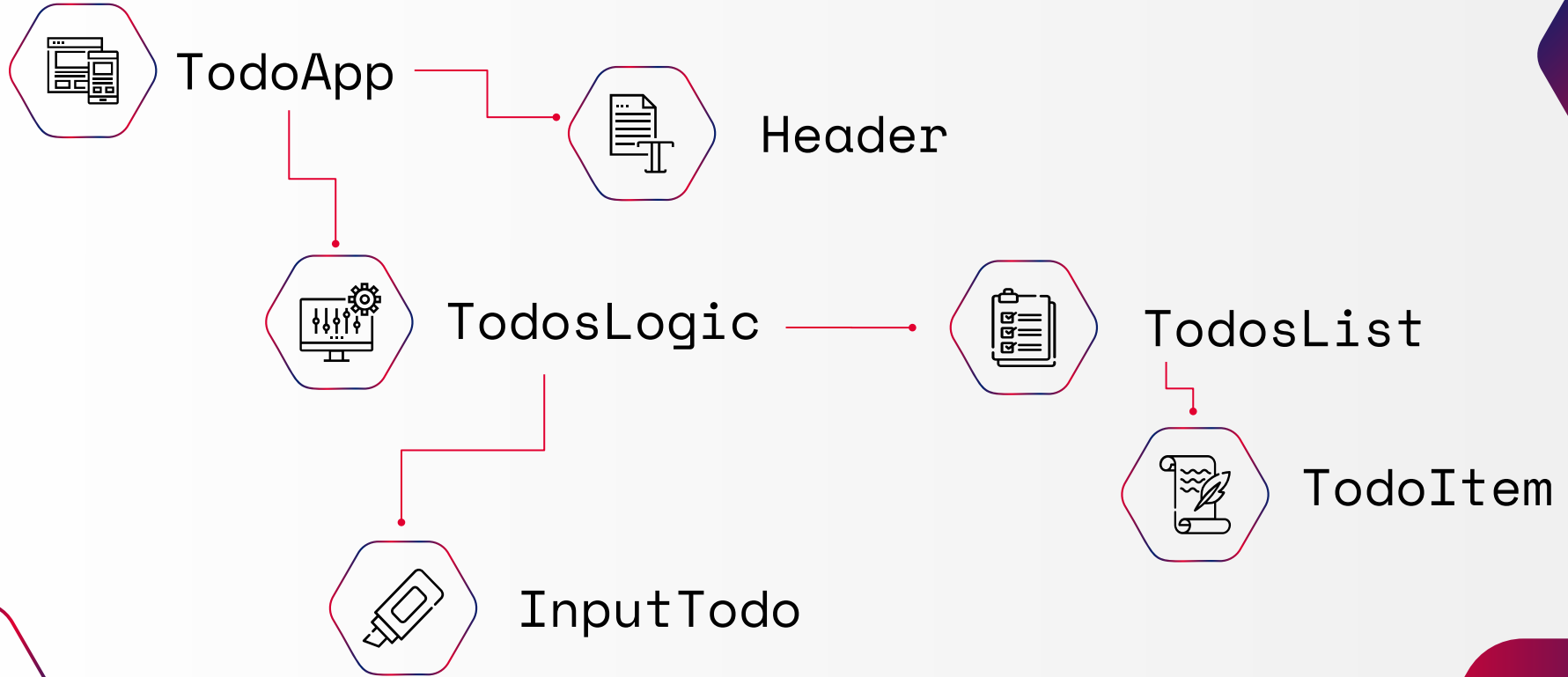
# Tworzenie Komponentów Projektu



## Podział:

1. **TodoApp** - komponent nadrzędny (główny).
2. **Header** - wyświetla tekst nagłówka.
3. **TodosLogic** - przechowuje logikę aplikacji.
4. **InputTodo** - pobierze dane wejściowe od użytkownika.
5. **TodosList** - pojemnik na elementy.
6. **TodoItem** - renderuje pojedynczy element todo.

# Układ Komponentów



Tworzymy **src/main.jsx** z następującym kodem:

```
import React from "react";  
import ReactDOM from "react-dom/client";  
  
import TodoApp from "./components/TodoApp";  
  
const domContainer = document.getElementById("root");  
const root = ReactDOM.createRoot(domContainer);  
root.render(<TodoApp />);
```

Tworzymy **src/components/ToDoApp.jsx** z kodem:

```
const ToDoApp = () => {  
  return (  
    <div>  
      <h1>Hello world!</h1>  
      <p>I am in a React Component!</p>  
    </div>  
  );  
};  
export default ToDoApp;
```

# Konwencja

- Nazwy komponentów zawsze piszemy wielką literą.
- Używanie PascalCase dla nazw plików składowych jest dobrą konwencją. Na przykład `TodoApp.jsx`.



# Obsługa Ścieżek Bezwzględnych

W **vite.config.js**:

```
import { defineConfig } from 'vite'  
import react from '@vitejs/plugin-react'
```

```
import path from 'path';  
export default defineConfig({  
  resolve: {  
    alias: {  
      '@': path.resolve(__dirname, './src'),  
    },  
  },  
  plugins: [react()],  
});
```

Tworzymy **jsconfig.json** z kodem:

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"]
    }
  },
}
```

Zmieniamy import w **src/main.jsx**:

```
import TodoApp from '@components/TodoApp';
```

# React Fragments

◁**React.Fragment**▷ lub skrót ◁> ◁/>

```
import React from 'react';
```

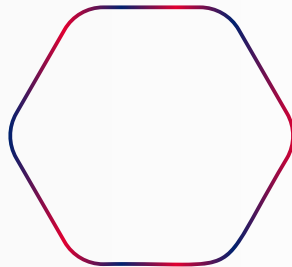
```
const TodoApp = () => {  
  return (  
    <React.Fragment>  
      <h1>Hello world!</h1>  
      <p>I am in a React Component!</p>  
    </React.Fragment>  
  );  
};  
export default TodoApp;
```

# Pozostałe komponenty

Umieszczone w **src/components**:

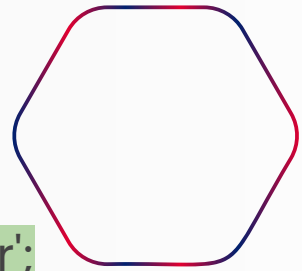
- Header.jsx
- TodosLogic.jsx
- InputTodo.jsx
- TodosList.jsx
- TodoItem.jsx

# Header



```
const Header = () => {  
  return (  
    <header>  
      <h1>todos</h1>  
      <p>Items will persist in the browser local storage</p>  
    </header>  
  );  
};  
export default Header;
```

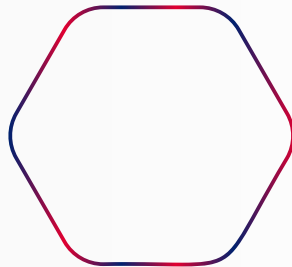
# Renderowanie Header'u



```
import Header from '@components/Header';
```

```
const TodoApp = () => {  
  return (  
    <>  
    <Header />  
    </>  
  );  
};  
export default TodoApp;
```

# TodosLogic



```
const TodosLogic = () => {  
  return (  
    <div>TodosLogic content</div>  
  )  
}  
export default TodosLogic;
```

## **Render TodosLogic in TodoApp:**

```
import TodosLogic from '@components/TodosLogic';  
...  
<TodosLogic />
```

# React Strict Mode



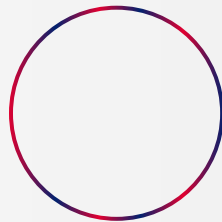
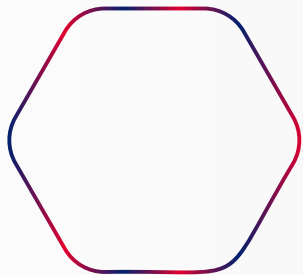
Strict Mode może pomóc zidentyfikować potencjalne problemy w aplikacji podczas opracowywania i może pokazywać komunikat ostrzegawczy w czasie wykonywania.

Możemy aktywować sprawdzanie dowolnej części lub całej aplikacji, dodając tag `<React.StrictMode>`.



Włączamy  
Strict Mode  
dla całej  
aplikacji,  
opakowując  
komponent  
główny -  
src/main.jsx:

```
root.render(  
  <React.StrictMode>  
    <TodoApp />  
  </React.StrictMode>  
);
```



# Model Danych React

W React każdy komponent ma tendencję do odbierania i przekazywania fragmentów informacji do innych komponentów w drzewie.

# React Props

Właściwości w React można uznać za atrybuty w elemencie HTML.

```
<input type="submit" value="Submit" />
```

Można użyć tych właściwości do przekazywania informacji, które zmieniają zachowanie elementu wejściowego.

W React dane przepływają od rodzica w dół drzewa komponentów. Kiedy komponent potomny otrzymuje właściwości od rodzica, to ta wartość staje się niezmienna i nie może być modyfikowana przez komponent odbierający.

# React State

Stan w React można traktować jako silnik napędowy aplikacji. W React stan deklarowany jest w komponencie, jeśli dane zmieniają się w czasie (zwykle przez interakcję użytkownika). Na przykład, jeśli komponent musi wykonywać interakcje użytkownika, takie jak aktualizowanie pola wejściowego lub zapamiętywanie poprzednich zdarzeń takich jak np. przełączanie przycisku.

W odróżnieniu od właściwości, stan w komponencie jest lokalny i specyficzny dla tego komponentu. Nie jest dostępny dla innych komponentów w drzewie, chyba że komponent zdecyduje się przekazać go do komponentu podrzędnego. Kiedy dane stanu są przekazywane do komponentu podrzędnego, są one przekazywane jako właściwość i tak właśnie są traktowane w odbierającym komponencie.

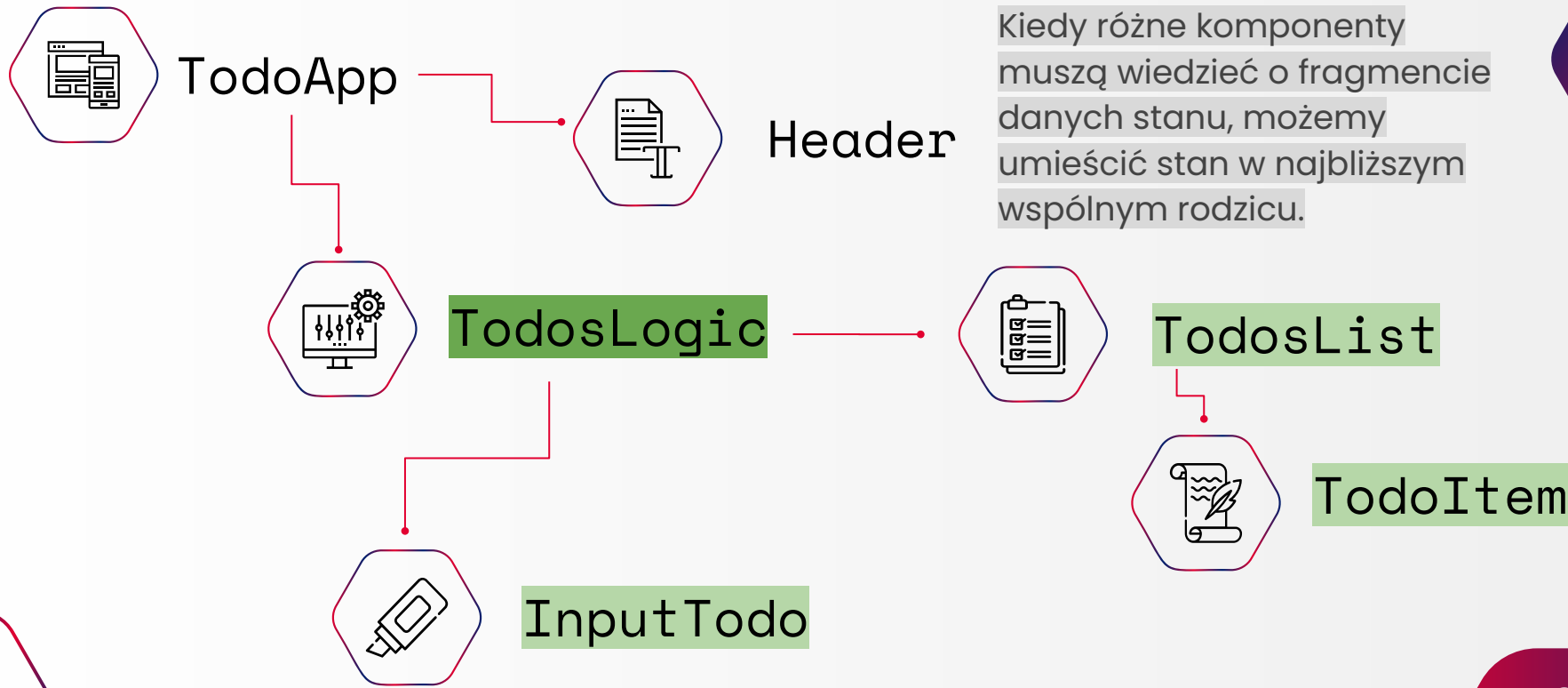
# Renderowanie Stanu

W aplikacji todos wszystko w interfejsie użytkownika to zbiór informacji. Te informacje również będą się zmieniać w czasie i dlatego będą przechowywane w stanie.

Pojedynczy wpis możemy przedstawić za pomocą obiektu, który wygląda tak:

```
{  
  id: 1,  
  title: "first todos item",  
  completed: false  
},  
// ...
```

# Gdzie Umieścić Stan



## Inicjowanie zmiennej todos w **components/TodosLogic.jsx**:

```
const TodosLogic = () => {  
  const todos = [  
    {  
      id: 1,  
      title: 'Setup development environment',  
      completed: true,  
    },  
    {  
      id: 2,  
      title: 'Develop website and add content',  
      completed: false,  
    },  
    {  
      id: 3,  
      title: 'Deploy to live server',  
      completed: false,  
    },  
  ],  
  return (  
    <ul>  
      {todos.map((todo) => (  
        <li>{todo.title}</li>  
      )))}  
    </ul>  
  )  
}  
export default TodosLogic;
```

## Renderowanie Komponentów **InputTodo** i **TodosList**:

```
const InputTodo = () => {  
  return <div>input field here...</div>;  
};  
export default InputTodo;
```

```
const TodosList = (props) => {  
  return (  
    <ul>  
      {props.todosProps.map((todo) => (  
        <li>{todo.title}</li>  
      ))}  
    </ul>  
  );  
};  
export default TodosList;
```



## Import do **TodosLogic**:

```
import InputTodo from '@components/InputTodo';  
import TodosList from '@components/TodosList';
```

```
const TodosLogic = () => {  
  // ...  
  return (  
    <div>  
      <InputTodo />  
      <TodosList todosProps={todos} />  
    </div>  
  );  
};  
export default TodosLogic;
```

## Destrukturyzacja Obiektu Props

```
const TodosList = (props) => {  
  const { todosProps } = props;  
  return (  
    <ul>  
      {todosProps.map((todo) => (  
        <li>{todo.title}</li>  
      ))}  
    </ul>  
  );  
};  
export default TodosList;
```

```
const TodosList = ({ todosProps }) => {  
  return (  
    <ul>  
      {todosProps.map((todo) => (  
        <li>{todo.title}</li>  
      ))}  
    </ul>  
  );  
};  
export default TodosList;
```

## Renderowanie Komponentu **TodoItem**

```
const TodoItem = ({ itemProp }) => {  
  return <li>{itemProp.title}</li>;  
};  
export default TodoItem;
```

```
import TodoItem from '@components/TodoItem';
```

```
const TodosList = ({ todosProps }) => {  
  return (  
    <ul>  
      {todosProps.map((todo) => (  
        <TodoItem itemProp={todo} />  
      ))}  
    </ul>  
  );  
};  
export default TodosList;
```

## Key Prop Dla Elementów Listy

```
import TodoItem from '@components/TodoItem';

const TodosList = ({ todosProps }) => {
  return (
    <ul>
      {todosProps.map((todo) => (
        <TodoItem key={todo.id} itemProp={todo} />
      ))}
    </ul>
  );
};

export default TodosList;
```

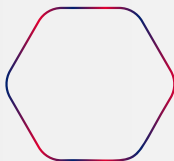


# Na podstawie:

<https://ibaslogic.com/react-components-data-model/#the-react-data-model>

# Koniec

Przygotował Adam Trentowski



**CREDITS:** This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**