Web Programming

User Interfaces

Prof. Josué Obregón
Department of Industrial Engineering- ITM
Seoul National University of Science and Technology

Objectives

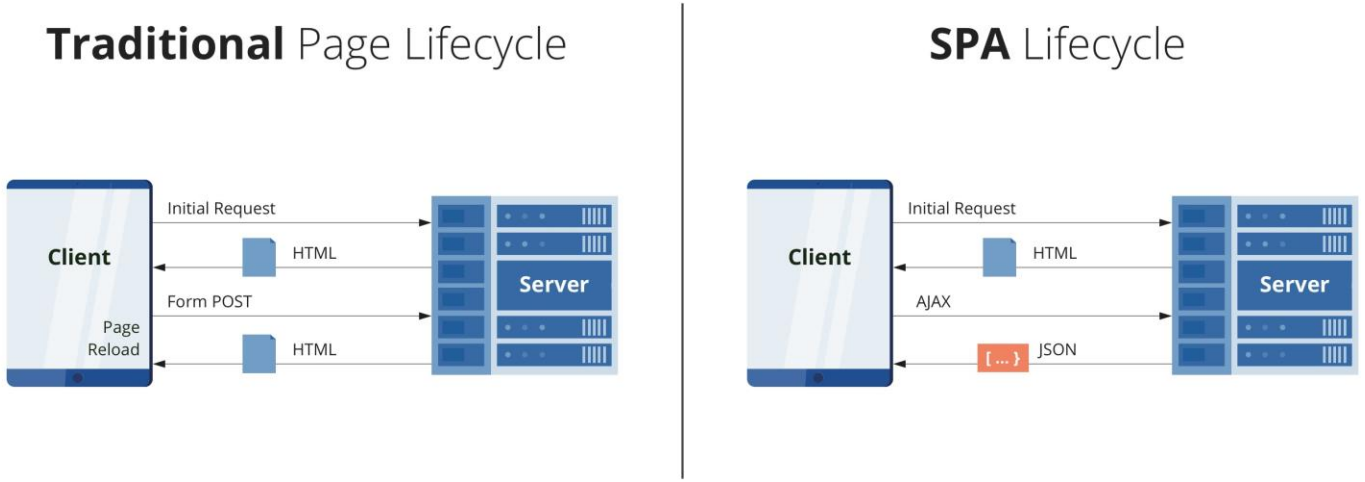
- Identify and describe common paradigms for designing user interfaces for web applications
- Learn React, a modern JavaScript library for creating dynamic web content and construct a simple dynamic web page.

Agenda

- Singe Page Applications
- React

Single-Page Application (SPA)

- Web application that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the loading entire new pages.
- The goal is faster transitions that make the website feel more like a native app.



Example https://www.airbnb.com/

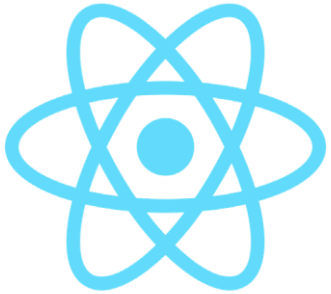
Image: https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli

SEOULTECH
SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

React

What is React?

- React is a **JavaScript framework**
- Used for **front end web development**
- Created and used by **Facebook**
- Famous for implementing a **virtual dom**



https://react.dev/learn

Fundamentals of React

1. JavaScript and HTML in the same file (JSX)
2. Declarative programming
3. React apps are made out of components.

JSX (JavaScript Syntax Extension or JavaScript XML)

- Is an extension to JavaScript.
- It provides an easier way to create UI components in React.
- Here's an example of its syntax:
  - `const element = <h1>Hello, World!</h1>;`
- With JSX, you can write code that looks very similar to HTML or XML, but you have the power of seamlessly mixing JavaScript methods and variables into your code.
- JSX is interpreted by a transpiler, such as Babel, and rendered to JavaScript code that the UI Framework (React, in this case) can understand.

JavaScript and HTML *in the same file*

Traditional approach

React approach

Declarative Programming

- A style of building the structure and elements of computer programs that expresses the logic of a computation without describing its control flow.
- Contrasted to imperative programming which expresses commands
  - uses statements that change a program's state.
- Declarative programming is a non-imperative style of programming in which programs describe their desired results without explicitly listing commands or steps that must be performed.

Imperative Programming

View

```
<h1>0</h1>
```

Logic

```
let num = parseInt(document.querySelector("h1").innerHTML);
num += 1;
document.querySelector("h1").innerHTML = num;
```

Declarative Programming

View

```
<h1>{num}</h1>
```

Logic

```
num += 1;
```

Components

- React apps are made out of components.
- A component is a piece of the UI (user interface) that has its own logic and appearance.
- A component can be as small as a button, or as large as an entire page.
- React components are JavaScript functions that return markup:

```
function MyButton() {
    return (
        <button>I'm a button</button>
    );
}
```

Components

Functions help break your code into small, reusable pieces

Components are functions for user interfaces

Math function:

Component function:

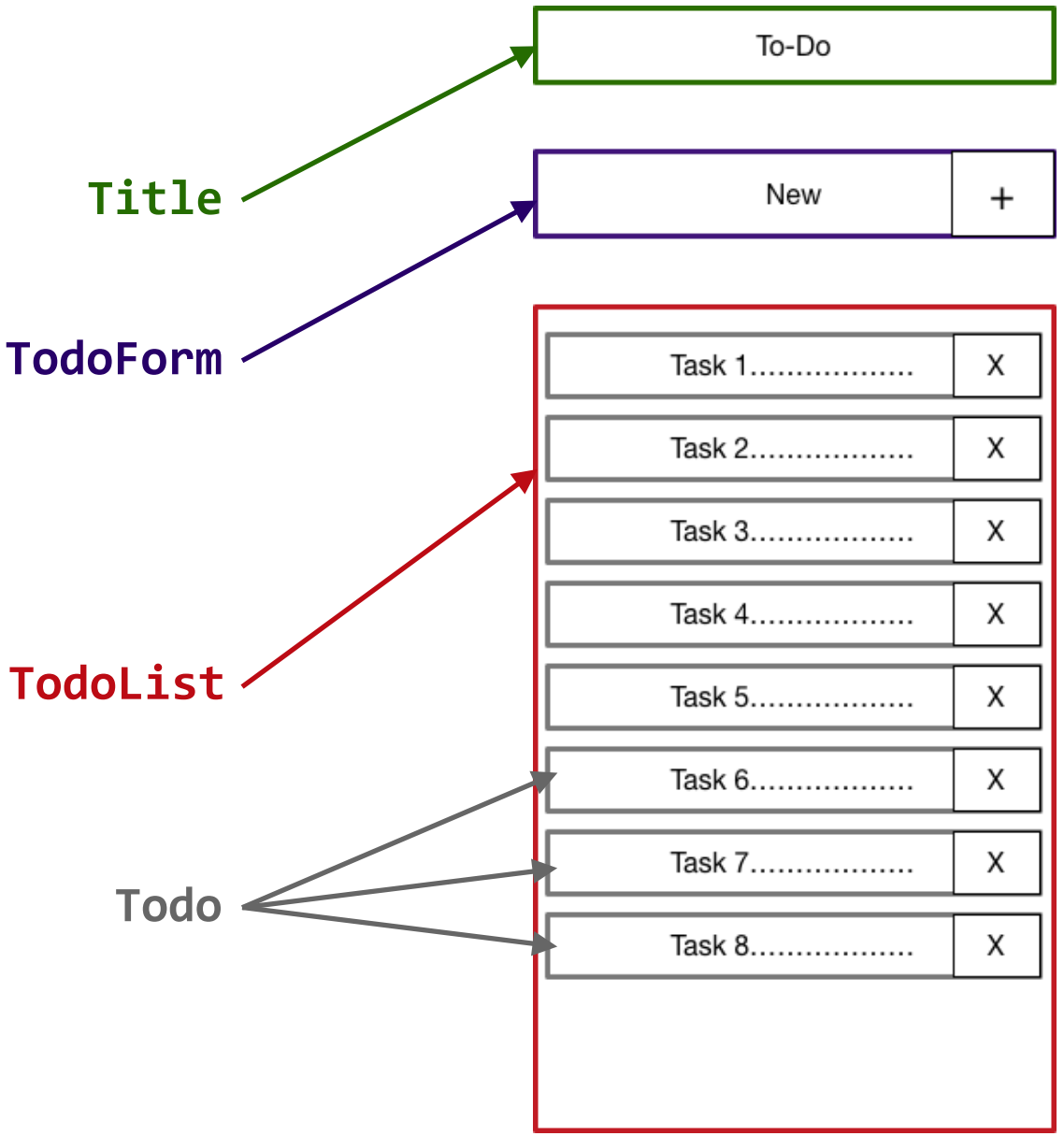
Todo application

Big idea:

- A digital to-do list

First step:

- mockup / wireframe



Required imports

- React
  - Base JavaScript library for building user interfaces
- ReactDOM
  - The react-dom/client APIs let you render React components on the client (in the browser). These APIs are typically used at the top level of your app to initialize your React tree. A framework may call them for you. Most of your components don't need to import or use them.
- Babel
  - Babel can convert JSX syntax into JavaScript

React Components

- React components are JavaScript functions that return markup
- After declaring MyButton, we can nest them into other components
  - Using the syntax <COMP_NAME />

```
function MyButton() {
    return (
        <button>I'm a button</button>
    );
}
```

```
MyApp() {
    return (
        <div>
            <h1>Welcome to my app</h1>
            <MyButton />
        </div>
    );
}
```

https://react.dev/reference/react-dom/components/common

Rendering Components

- Let's say there is a somewhere in your HTML file

```
<div id="app"></div>
```

- We call this a "root" DOM node because everything inside it will be managed by React DOM.
  - Applications built with just React usually have a single root DOM node.
  - If you are integrating React into an existing app, you may have as many isolated root DOM nodes as you like.
- To render a React element:

```
const root = createRoot(document.querySelector('#app'));
root.render(<App />);
```

React Props

- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.
- Props stands for properties.

```
const myElement = <Car brand="Ford" />;

function Car(props) {
    return <h2>I am a {props.brand}!</h2>;
}
```

React Hooks

- Hooks allow function components to have access to state and other React features.
- Hooks let you use different React features from your components.
- Hooks allow us to "hook" into React features such as state and lifecycle methods.
- You must import Hooks from react.
- Here we are using the useState Hook to add state to a component

```
const [color, setColor] = useState("red");
```

Responding to Events

- React lets you add event handlers to your JSX.
- React events are named using camelCase, rather than lowercase.
- React events are also known synthetic event which are cross-browser wrappers around the browser's native event
- To add an event handler, you will first define a function and then pass it as a prop to the appropriate JSX tag.

```
function Button() {
    function handleClick() {
        alert('You clicked me!');
    }

    return (
        <button onClick={handleClick}>
            Click me
        </button>
    );
}
```

https://react.dev/learn/responding-to-events

Web Programming

Some of the contents of this slide are based or adapted from CS50web course. Attribution is given to the creators of this course. CS50web course is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.