React Core Concepts

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	1	**Week No:**	09	**Semester:**	
Lecturer:	*Sazzad Hossain; sazzad@aiub.edu*				

Lecture Outline

- ✓ React Core Concepts
- ✓ Functional components
- ✓ Props
- ✓ State
- ✓ Hooks
- ✓ JSX
- ✓ React Events

React Core Concepts

Following are the core ReactJS component That uses in NextJS

- **Functional Components**
- **State and Props**
- **React Hooks**
- **JSX**

Functional components

Functional components are the simplest type of React components. They are JavaScript functions that **return** JSX, describing what should be **rendered** on the screen. They do not have state or lifecycle methods, making them lightweight and easy to understand.

```
import React from 'react';

const FunctionalComponent () {
  return <h1>Hello, I am a functional
component!</h1>;
};
```

Props

Props (short for "properties") are a mechanism for **passing data** from a **parent** component to a **child** component. They allow you to customize and configure child components based on values provided by the parent component. Props are read-only and cannot be modified by the child component.

Props

```
//ParentComponent.js
import React from 'react';
import ChildComponent from './ChildComponent';

const ParentComponent = () => {
  // Define props data
  const name = 'John Doe';
  const age = 30;

  return (
    <div>
      {/* Pass props to the ChildComponent */}
      <ChildComponent name={name} age={age} />
    </div>
  );
};

export default ParentComponent;
```

```
//ChildComponent.js
import React from 'react';

const ChildComponent = (props) => {
  // Access props data
  const { name, age } = {props.name,props.age};

  return (
    <div>
      <h1>Hello, {name}!</h1>
      <p>You are {age} years old.</p>
    </div>
  );
};

export default ChildComponent;
```

State

State is a built-in feature that allows **you to manage and store data** within a component. It represents the **mutable data that can change over time** and affects how the component **renders** and **behaves**. When state data is updated, React automatically re-renders the component to reflect the changes.
State is a **fundamental concept** in React that enables dynamic and interactive user interfaces. It allows components to **maintain their own local data and respond to user interactions,** making React applications more interactive and responsive.

Hooks

- React hooks are functions introduced in React 16.8 that allow functional components to have **state** and use **lifecycle methods** without the need for writing class components.
- Hooks provide a more **straightforward and reusable way to manage state and perform side effects** in functional components.

Hooks

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

export default function CounterComponent = () => {
  // useState hook to manage state
  const [count, setCount] = useState(0);

  // useEffect hook to perform side effects
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);

  // Function to handle the increment button click
  const handleIncrement = () => {
    setCount(count + 1);
  };

  // Function to handle the decrement button click
  const handleDecrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <h1>Counter: {count}</h1>
      <button onClick={handleIncrement}>Increment</button>
      <button onClick={handleDecrement}>Decrement</button>
    </div>
  );
};
```

Hooks

useState: The useState hook allows us to **add state to functional components.** We declare a **state variable** called count and its **updater function setCount** by calling useState(0) with an initial value of 0. We can now manage the state of count using setCount.

useEffect: The useEffect hook enables us to perform **side effects in functional components**. In this example, we use it to update the document title with the current count value. The **useEffect hook takes a function as its first argument**, and the **second argument is an array of dependencies that specify when the effect should run**. In this case, we pass [count] as the dependency array, so the effect will only run when the count state changes.

JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used with React to describe the structure of UI components. It allows developers to **write HTML-like code within JavaScript**, making it easier to create and visualize the component's UI.

JSX

```
import React from 'react';

export default const GreetingComponent = () => {
  const name = 'John Doe';
  const showGreeting = true;

  return (
    <div>
      {showGreeting ? (
        <h1>Hello, {name}!</h1>
      ) : (
        <p>No greeting available.</p>
      )}
    </div>
  );
};
```

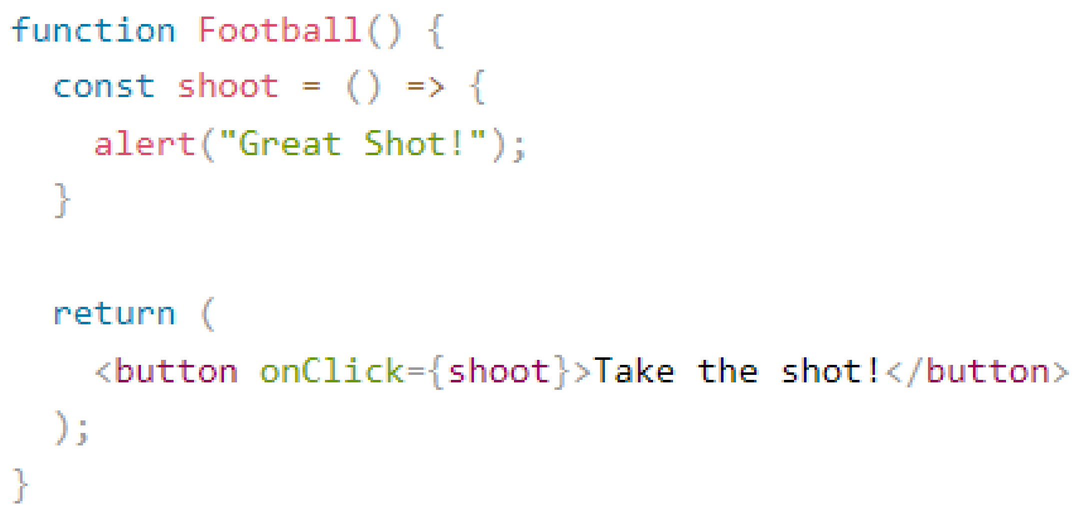
React Events

Just like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

React events are written in **camelCase** syntax:
`onClick` instead of `onclick`.
React event handlers are written inside curly braces:
`onClick={shoot}` instead of `onClick="shoot()"`.
There are many events supported by React. Below are some popular events;

React Events



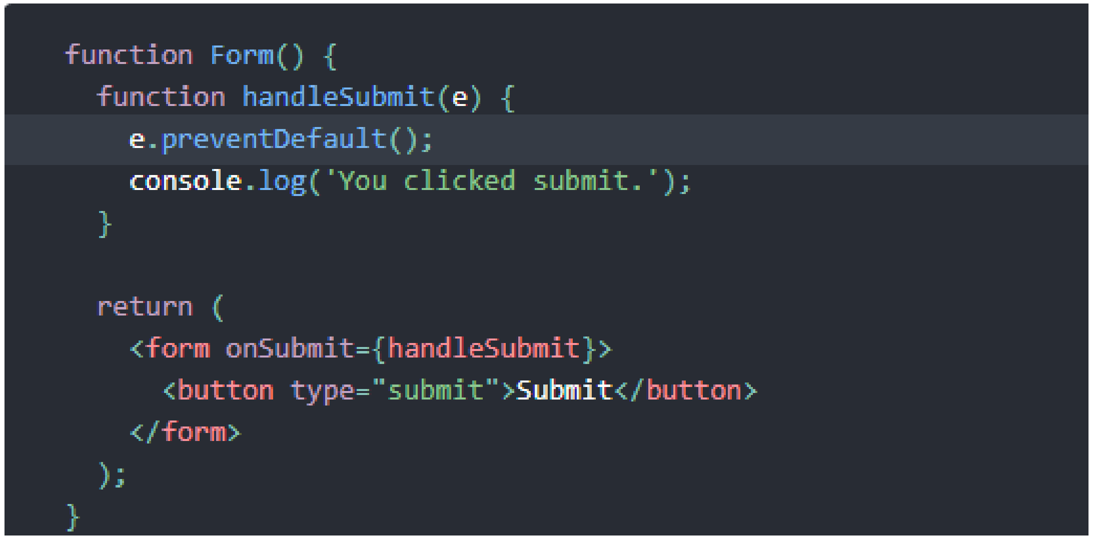
```
function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );
}
```

https://www.w3schools.com/react/showreact.asp?filename=demo2_react_events_handler

React Events

- **Keyboard Events**
- **Focus Events**
- **Form Events**
- **Generic Events**
- **Mouse Events**
- **Pointer Events**
- **Selection Events**
- **Touch Events**
- **UI Events**
- **Wheel Events**
- **Media Events**
- **Image Events**

React Events



```
function Form() {
  function handleSubmit(e) {
    e.preventDefault();
    console.log('You clicked submit.');
  }

  return (
    <form onSubmit={handleSubmit}>
      <button type="submit">Submit</button>
    </form>
  );
}
```

e is a synthetic event.

The **preventDefault()** method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

References

1. **W3Schools Online Web Tutorials, URL: http://www.w3schools.com**
2. **Next.js, URL:** https://nextjs.org/
3. **Mozilla Developer Networks, URL:** https://developer.mozilla.org/

Thank You!