24 November 2024 GIAIC Class Summary

All codes of class are available here:

https://github.com/iamahsanirfan/24-Nov-Nextjs-Class-Code-GIAIC

---

1. Responsive (through Grid)

Responsiveness ensures that a web application looks good and functions well on devices of all sizes, from mobile phones to large monitors.

The **Grid system** is a layout mechanism in CSS (and frameworks like Tailwind CSS or Bootstrap) that divides the webpage into rows and columns. This approach allows you to place elements flexibly while keeping them aligned.

In Next.js (or React), you might use libraries like **Tailwind CSS** to implement grids. For example:

```
<div className="grid grid-cols-3 gap-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

- `grid-cols-3`: Divides the container into three columns.
- `gap-4`: Adds spacing between items.

Responsive designs use media queries or utilities to adapt layouts:

```
<div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
  <div>Item 4</div>
</div>
```

- This layout shows **1 column on small screens**, **2 on medium screens**, and **4 on large screens**.

---

2. State or `useState()`

State refers to data that determines how your app looks or behaves at any given time. In React, `useState()` is a hook that allows you to add state to functional components.

Example:

```
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0); // Declare state variable

  const increment = () => setCount(count + 1); // Update state

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increase</button>
    </div>
  );
}
```

- `count`: Holds the current state value.
- `setCount`: Function to update the state.
- Clicking the button increases the count by 1.

---

3. React Hook

Hooks are functions introduced in React 16.8 that allow you to use React features (like state and lifecycle methods) in functional components instead of class components.

Common hooks:

- `useState`: For managing state.
- `useEffect`: For performing side effects like fetching data or updating the DOM.
- `useContext`: For accessing context values across components.

Hooks follow rules:

1. Call them at the top level (not inside loops, conditions, or nested functions).
2. Use them only in React functional components.

---

4. Event Handler

Event handlers in React let you define what happens when a user interacts with your app (e.g., clicking a button, typing in a field).

Example:

```
function ClickMe() {
  const handleClick = () => alert("Button clicked!");

  return <button onClick={handleClick}>Click Me</button>;
}
```

- **`onClick`**: Specifies the event.
- **`handleClick`**: Function executed when the button is clicked.

Common events:

- **`onClick`**: For clicks.
- **`onChange`**: For input changes.
- **`onSubmit`**: For form submissions.

---

5. Props

Props (short for properties) are used to pass data from one component to another in React.

Example:

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}

function App() {
  return <Greeting name="John" />;
}
```

- **`name="John"`**: Prop passed to the `Greeting` component.
- **`props.name`**: Accesses the value of the `name` prop in `Greeting`.

Props are **read-only**, meaning the receiving component cannot modify them directly.

---

6. What is `<></>`?

The `<> </>` syntax represents a React **Fragment**. Fragments allow you to group multiple elements without adding extra nodes (like a `div`) to the DOM.

Example:

```
function List() {
  return (
    <>
      <h1>Item 1</h1>
      <h1>Item 2</h1>
    </>
  );
}
```

- No unnecessary `div` wrapping the `<h1>` elements.
- Improves performance and keeps the DOM clean.

Let me know if you'd like more examples or details!