### 1. ****Introduction to Next.js Layout and Components****

* **Objective**: Introduce what layout and components are in Next.js.
* **Key Points**:
  + **Next.js Layout**: Structure that’s consistent across multiple pages (e.g., header, footer).
  + **Components**: Reusable building blocks of UI (e.g., buttons, cards).
* **Demo/Example**: Explain the basic structure of a Next.js project.

### 2. ****Setting Up a Layout in Next.js****

* **Objective**: Show how to create a layout component and use it across multiple pages.
* **Key Points**:
  + Create a new Layout.js file inside the components folder.
  + Layout includes header, footer, and main content.
  + Wrap every page with the layout component.

jsx

// components/Layout.jsexport default function Layout({ children }) {

return (

<div>

<header>

<h1>My Website</h1>

</header>

<main>{children}</main>

<footer>

<p>&copy; 2024 My Website</p>

</footer>

</div>

);

}

In the pages (like index.js or about.js), use the layout:

jsx

import Layout from '../components/Layout';

export default function Home() {

return (

<Layout>

<h2>Welcome to Home Page</h2>

<p>This is the home page content.</p>

</Layout>

);

}

### 3. ****Adding Global Styles with Layout****

* **Objective**: How to add global CSS to the layout.
* **Key Points**:
  + Import global CSS files to make sure styles apply across all pages.
  + Demonstrate how to style the layout (header, footer).

jsx

// styles/globals.css

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

box-sizing: border-box;

}

header, footer {

background-color: #333;

color: white;

text-align: center;

padding: 1rem;

}

main {

padding: 2rem;

}

Import the CSS in \_app.js:

jsx

import '../styles/globals.css';

function MyApp({ Component, pageProps }) {

return <Component {...pageProps} />;

}

export default MyApp;

### 4. ****Creating Reusable Components in Next.js****

* **Objective**: Explain how to create simple and reusable components.
* **Key Points**:
  + Create separate files for each component inside the components folder.
  + Show how components are independent and reusable.

jsx

// components/Button.jsexport default function Button({ label, onClick }) {

return (

<button onClick={onClick} style={{ padding: '10px 20px', background: '#0070f3', color: 'white', border: 'none' }}>

{label}

</button>

);

}

Use the Button component in a page:

jsx

import Layout from '../components/Layout';

import Button from '../components/Button';

export default function About() {

const handleClick = () => {

alert('Button Clicked!');

};

return (

<Layout>

<h2>About Us</h2>

<p>Learn more about us on this page.</p>

<Button label="Click Me" onClick={handleClick} />

</Layout>

);

}

### 5. ****Passing Props to Components****

* **Objective**: Explain how to pass dynamic data to components using props.
* **Key Points**:
  + Show how data is passed from parent components to child components.
  + Example with a dynamic card component.

jsx

// components/Card.jsexport default function Card({ title, description }) {

return (

<div style={{ border: '1px solid #ddd', padding: '20px', marginBottom: '10px' }}>

<h3>{title}</h3>

<p>{description}</p>

</div>

);

}

Use Card component in a page:

jsx

import Layout from '../components/Layout';

import Card from '../components/Card';

export default function Services() {

return (

<Layout>

<h2>Our Services</h2>

<Card title="Web Development" description="We offer full-stack web development services." />

<Card title="App Development" description="We create modern mobile applications." />

</Layout>

);

}

### 6. ****Creating Dynamic Layout for Multiple Pages****

* **Objective**: Create a layout that adapts for different page needs.
* **Key Points**:
  + Demonstrate how to create conditional layouts (e.g., different headers for different pages).
  + Show dynamic use of layout based on page type.

jsx

// components/Layout.jsexport default function Layout({ children, pageTitle }) {

return (

<div>

<header>

<h1>{pageTitle || "My Website"}</h1>

</header>

<main>{children}</main>

<footer>

<p>&copy; 2024 My Website</p>

</footer>

</div>

);

}

Pass the title dynamically in the pages:

jsx

import Layout from '../components/Layout';

export default function Home() {

return (

<Layout pageTitle="Home Page">

<h2>Welcome to the Home Page</h2>

</Layout>

);

}

### 7. ****Adding Interactivity with Components****

* **Objective**: Show how to add event handlers like clicks, form submissions, etc.
* **Key Points**:
  + Demonstrate how to make components interactive using onClick, onChange, etc.
  + Example with form handling.

jsx

// components/Form.jsimport { useState } from 'react';

export default function Form() {

const [inputValue, setInputValue] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

alert(`Submitted: ${inputValue}`);

};

return (

<form onSubmit={handleSubmit}>

<input

type="text"

value={inputValue}

onChange={(e) => setInputValue(e.target.value)}

placeholder="Enter something"

/>

<button type="submit">Submit</button>

</form>

);

}

Use Form component:

jsx

import Layout from '../components/Layout';

import Form from '../components/Form';

export default function Contact() {

return (

<Layout>

<h2>Contact Us</h2>

<Form />

</Layout>

);

}

### 8. ****Conclusion****

* **Objective**: Wrap up the video with key takeaways.
* **Key Points**:
  + Recap the importance of layouts and components in Next.js.
  + Encourage reusability and modularity for better development practices.