vivasoft

Learnathon 2.0 - React Class 3

...

Learnathon - 2.0

Fardin Islam | Software Engineer (L-1) | Vivasoft Limited

vivasoft

Styling in React

- ❖ Global style file
- ❖ CSS modules
- ❖ Inline styles
- ❖ SASS/SCSS
- ❖ CSS-in-JS (Styled-component)
- ❖ CSS Framework (Tailwind CSS)

Learnathon - 2.0

Styling in React

- **Global style file**

```
//index.css
.counter {
  background-color: red;
  color: white;
  padding: 10px;
}
//app.jsx
return (
<div className="counter">
    Counter
</div>);
```

- **CSS modules**

```
//index.module.css
.counter {
background-color: red;
color: white;
padding: 10px;
}
//app.jsx
import style from './index.module.css'
return (
<div className={styles.counter}>
    Counter
</div>);
```

- **Inline styles**

```
//app.jsx
const styles = {
backgroundColor: "red",
color: "white",
padding: "10px",
};
return (
<div style={styles}>
    Counter
</div>);
```

Learnathon - 2.0

vivasoft

Styling in React

"CSS with superpowers"

❖ **CSS**

```
//index.css
nav ul {
margin: 0;
padding: 0;
list-style: none;
}
nav li {
display: inline-block;
}
```

❖ **SCSS**

```
//index.scss
nav {
  ul {
      margin: 0;
      padding: 0;
      list-style: none;
      }
  li {
      display: block;
      }
}
```

❖ **SASS**

```
//app.sass
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block
```

Learnathon - 2.0

vivasoft

Styling in React

- **CSS-in-JS (Styled-component)**

```
//app.jsx
const Container = styled.div`
     background-color: red;
     color: white;
     padding: 10px;
`;
return <Container>Counter</Container>;
```

- **CSS Framework (Tailwind CSS)**

```
//app.jsx
return
<div className="bg-red-500 text-white p-2">
     <button className="bg-yellow-500">
          Counter
     </button>
</div>;
```

Learnathon - 2.0

vivasoft

Dynamic Routing in React

```
const routes = createBrowserRouter([
{
    path: "/",
    element: <App />,
    children: [{ path: "counter", element: <Counter /> }],
},
{
    path: "/users:id",
    element: <Counter />,
},
]);
ReactDOM.createRoot(document.getElementById("root")).render(
<React.StrictMode> <RouterProvider router={routes} /></React.StrictMode>
);
```

Learnathon - 2.0

vivasoft

React Hooks

- useState
- useEffect
- useReducer
- useContext
- useMemo
- useCallback
- useRef

Learnathon - 2.0

vivasoft

React Hooks(useState)

```
import React, { useState } from 'react';
function ExampleComponent() {
  const [count, setCount] = useState(0);
  const handleClick = () => setCount(count + 1);
return (
      <div>
        <p>Count: {count}</p>
        <button onClick={handleClick}>Increment</button>
      </div>
    );
}
export default ExampleComponent;
```

- **Adding state**
- **Updating state**
- **Updating objects and arrays**

Learnathon - 2.0

vivasoft

React Hooks(useEffect)

```
useEffect(() => {
  console.log("i am in use effect!")
  return () => {
    console.log("This is my clean up")
  }
},[]) //dependency array
```

- **Fetching data**
- **Updating the DOM**
- **Conditional Effects**
- **Cleaning Up**
- **Effects in custom Hooks**

Learnathon - 2.0

vivasoft

React Hooks(useReducer)

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

function reducer(state, action) {
// ...
}
function MyComponent() {
const [state, dispatch] = useReducer(reducer,initialArg);
// ...
}
```

- **Complex UI State**
- **Global State Management**
- **Custom Logic and State Transitions**

Learnathon - 2.0

vivasoft

React Hooks(useContext)

```
const ThemeContext = createContext(null);
export default function MyApp() {
  return (
    <ThemeContext.Provider value="dark">
      <Button />
    </ThemeContext.Provider>
  )
}
function Button() {
const theme = useContext(ThemeContext);
// ...
}
```

- ❖ **Complex UI State**
- ❖ **Global State Management**
- ❖ **Optimizing re-renders**

Learnathon - 2.0

vivasoft

React Hooks(useCallback)

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

function MyComponent() {
// Define a callback function using useCallback
const increment = useCallback(() => {
     setCount(count + 1);
}, [count]);
// ...
}
```

- Skipping re-rendering
- Memoized callback function
- Optimizing a custom Hook

Learnathon - 2.0

vivasoft

React Hooks(useMemo)

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

function MyComponent({ count}) {
    const visibleTodos = useMemo(() =>
      costlyMathFun(count), [count]);
// ...
}
```

- ❖ **Skipping re-rendering**
- ❖ **Skipping expensive recalculations**
- ❖ **Memoizing a dependency**
- ❖ **Memoizing a function**

Learnathon 2.0

vivasoft

React Hooks(useRef)

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

function MyComponent() {
     const intervalRef = useRef(0);
     const inputRef = useRef(null);
// ...
}
```

- ❖ **Referencing a value**
- ❖ **Manipulating the DOM**
- ❖ **Keeping Track without rerendering**

Learnathon - 2.0

vivasoft

Higher Order Component (HOC)

"A higher-order component is a function that takes a component and returns a new component."

- ❖ HOC doesn't modify the input component.
- ❖ HOC is a pure function with zero side-effects.
- ❖ HOC composes the original component by wrapping it

Learnathon - 2.0

vivasoft

Higher Order Component (HOC)

```
//Higher_Order_Component
const withFuntionality = (InputComponent) => {
const NewComponent =(props) =>{
// HOC_specific_Functionality
     return <InputComponent {...props} newProp="value" />;
     };
   return NewComponent
};
export default withFuntionality;

const CounterComponent=()=> {
//...CounterComponent code
}
export default withFuntionality(CounterComponent);
```

Learnathon - 2.0

vivasoft

Render props

```
const WithFuntionality = (props) => {
   const [value,setValue] =useState(0)
   const handleValue =()=>setValue(prev=>prev+1)
   return props.render(value, handleValue)
};
export default WithFuntionality;
const MyApp=()=> {
   return(
     <WithFuntionality render=((value,handleValue)=>
       <CounterComponent count={value} handleCount={handleValue}/>
     />
     )
}
```

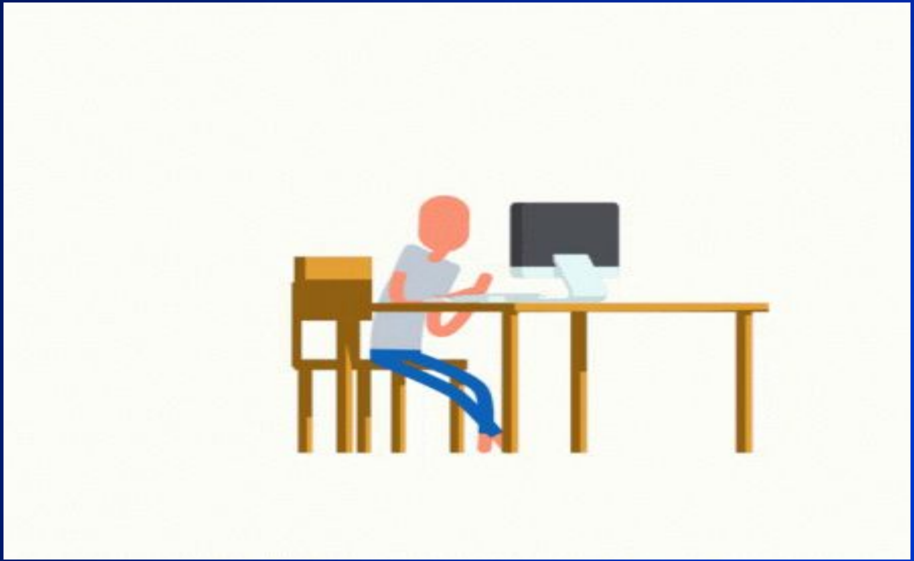
Learnathon - 2.0

vivasoft

Q&A time

Learnathon - 2.0

vivasoft



Thank You! Goodbye Everyone!

Learnathon - 2.0