Introduction to ReactJS

What is ReactJS?

- ReactJS adalah **JavaScript Library** yang digunakan untuk membangun antarmuka pengguna yang interaktif dan efisien.
- ReactJS dikembangkan oleh **Facebook** dan dirilis pada tahun 2013.
- ReactJS menggunakan **Virtual DOM** yang memungkinkan pengguna untuk membuat UI yang dinamis dan interaktif.

> Virtual DOM adalah representasi DOM yang disimpan di memori dan disinkronkan dengan DOM yang sebenarnya oleh pustaka seperti ReactDOM.

Why React?

- **Declarative**: React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.
- **Component-Based**: Build encapsulated components that manage their own state, then compose them to make complex UIs.
- **Learn Once, Write Anywhere**: Develop new features in React without rewriting existing code. React can also render on the server using Node and power mobile apps using React Native.

Key Concepts in React

1. **JSX**: A syntax extension for JavaScript recommended for use with React to describe what the UI should look like.
2. **Components**: Independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render function.
3. **Props**: Short for properties, these are read-only components that must be kept pure.
4. **State**: Allows components to create and manage their own data.

React Ecosystem

- **React Router**: Declarative routing for React.
- **Redux**: A predictable state container for JavaScript apps.
- **Next.js**: A minimalistic framework for server-rendered React applications.

JSX in Detail

- **JSX**: A syntax that allows HTML and JavaScript to coexist within the same file.
- **Why JSX?**: It makes it easier to write and add HTML in React.

Components

Functional Components

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

Class Components

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>
  }
}
```

Components

Komponen Fungsional

- **Definisi**: Komponen fungsional adalah komponen yang ditulis sebagai fungsi JavaScript. Mereka menerima props sebagai argumen dan mengembalikan elemen React.
- **Sederhana dan Ringkas**: Biasanya lebih ringkas dan lebih mudah dibaca, terutama untuk komponen yang lebih kecil dan fokus pada UI.
- **Hooks**: Dengan diperkenalkannya Hooks dalam React 16.8, komponen fungsional dapat menggunakan state dan fitur React lainnya tanpa menulis sebuah kelas.

Components

Komponen Kelas

- **Definisi**: Komponen kelas adalah komponen yang lebih tradisional dalam React. Mereka ditulis dengan menggunakan kelas JavaScript.
- **State dan Lifecycle**: Komponen kelas memungkinkan penggunaan state dan lifecycle methods secara langsung dalam komponen.
- **Lebih Kompleks**: Mereka cenderung lebih kompleks dan memiliki lebih banyak fitur dibandingkan dengan komponen fungsional, meskipun Hooks telah mengurangi perbedaan ini.

Components

State

- **Komponen Kelas**: Dalam komponen kelas, state adalah objek yang menyimpan informasi yang mungkin berubah sepanjang waktu komponen tersebut hidup. State ini diinisialisasi dalam konstruktor dan diubah dengan menggunakan metode `setState`.
- **Komponen Fungsional**: Sebelum Hooks, komponen fungsional tidak bisa menggunakan state. Namun, dengan `useState`, state kini dapat digunakan dalam komponen fungsional.

Components

Hooks

- **useState**: Hook ini memungkinkan Anda menambahkan state React ke komponen fungsional.
- **useEffect**: Digunakan untuk melakukan efek samping dalam komponen fungsional, seperti operasi I/O, manipulasi data, dan lainnya. Ini mirip dengan lifecycle methods `componentDidMount`, `componentDidUpdate`, dan `componentWillUnmount` dalam komponen kelas.
- **Custom Hooks**: React juga memungkinkan pembuatan Custom Hooks, yang memperluas kemampuan komponen fungsional dan memungkinkan pembagian logika stateful antara berbagai komponen.

Props vs. State

Props:

- Passed to the component (similar to function parameters)
- Immutable (cannot be modified by the component)

State:

- Local to the component (similar to variables declared within a function)
- Mutable (can be modified by the component)

Props

Passing Props:

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

const element = <Welcome name="Sara" />
```

State

Setting State:

```
class Clock extends React.Component {
  constructor(props) {
    super(props)
    this.state = { date: new Date() }
  }
}
```

Hooks

useState:

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

function Example() {
  const [count, setCount] = useState(0)

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click Me</button>
    </div>
  )
}
```

State dan Hooks

Kesimpulan

- **Pilih Sesuai Kebutuhan**: Baik komponen fungsional maupun kelas memiliki tempatnya masing-masing dalam pengembangan React. Pilihan antara keduanya sering bergantung pada preferensi pribadi, kompleksitas komponen, dan fitur yang diperlukan.
- **Hooks Menambah Fleksibilitas**: Dengan Hooks, React telah memberikan lebih banyak kekuatan dan fleksibilitas kepada komponen fungsional, membuatnya mampu melakukan hampir semua yang bisa dilakukan oleh komponen kelas.

Dengan penjelasan ini, Anda dapat lebih memahami perbedaan antara komponen fungsional dan kelas di React, serta bagaimana state dan hooks berperan dalam kedua jenis komponen tersebut.