React Component

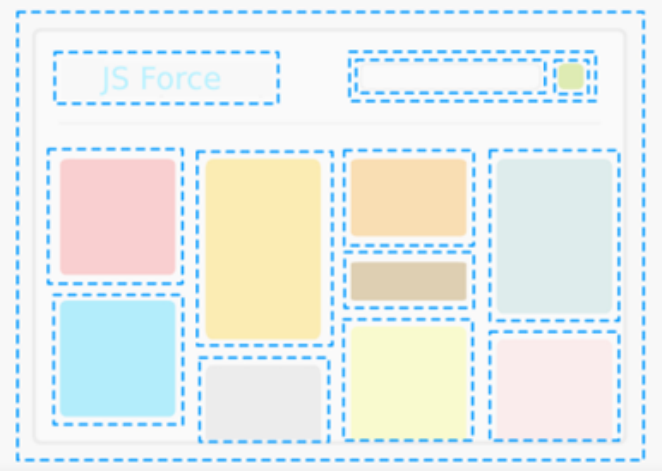
Topics covered:

- What are React components?
- Component based architecture
    - ❖ Types of react components
- Component rendering
    - ❖ Virtual Dom
    - ❖ Folder structure
    - ❖ rendering of component

1. React components:

- Components are one of React's fundamental building blocks. To put it another way, every application you create with React will be built up of what are known as components. Using components makes creating user interfaces considerably simpler. You can see a user interface (UI) divided into numerous separate parts, or components, and work on them independently before merging them all into a parent component to create your final UI.

Reusable components based ui



2. Component based architecture:

A component-based design divides a web page or web application into its smallest parts or units, allowing us to reuse it repeatedly without having to write the same code twice.

React offers two different kinds of components:

- ➔ Class-based components
- ➔ Function-based components

React components come with a built-in state object. When a component renders a component, persistent assets are stored in a state, which is encapsulated data.

When a user interacts with our application and changes its state, the UI may then appear entirely different. This is because the newly created state is used to represent that change, not the previous one. The process of initializing, updating, and altering a state's behavior in accordance with requirements is called State management.

- **Types of React components**
  - ❖ **Class-based components:**
    - ➔ A class is a particular kind of component in React that enables state management.
    - ➔ Until Hooks were introduced in React 16.8, the only way to use and maintain states was through a class-based component.
    - ➔ This is the reason why the only stateful component available at the time was a class.
    - ➔ Classes let us handle and alter the state in an efficient manner.

      A common class-based component contains the following:

      - For developing JSX, a React component was imported from the React library.
      - Classes can be created using this component imported from the React library.
      - A class component that extends the React Component.
      - To initialize the props, we need a constructor.
      - To paint/create the UI, we need a render method.
      - An export with the same name as the class.
      - The class component in this sample is named Demo, and it creates a header called Hello Everyone!

➔ Example:

```
import React, { Component } from 'react'
export default class Demo extends Component {
    constructor(props){
          super(props)
           }
  render() {
    return (
      <div>
          <h1>Hello Everyone!</h1>
      </div>
    )
  }
}
```

❖ **Function-based component:**

- A functional component is like a JavaScript function.
- It begins with the term "function", then comes a name enclosed in parentheses and curly braces.
- Functional components did not offer state management prior to React Hooks' introduction in version 16.5 of React.
- Consequently, stateless components were referred to as functional components.
- With the creation of Hooks, we can essentially perform everything that a class component can do, but faster and with less lines of code.
  A common function-based component contains the following:

    - For developing JSX, a React component was loaded from the Reacting library.
    - A function declaration should begin with a capital letter, followed by the function name.
    - Between parentheses, a parameter is passed; in React, this parameter is called props.
    - A return method has JSX in it.
    - An export with the same name as the function.
    - The ES6 syntax can also be used to create functional components, such as arrow functions. The function keyword is removed, but everything else is the same.
    - There is only one container we can return. For instance, in this

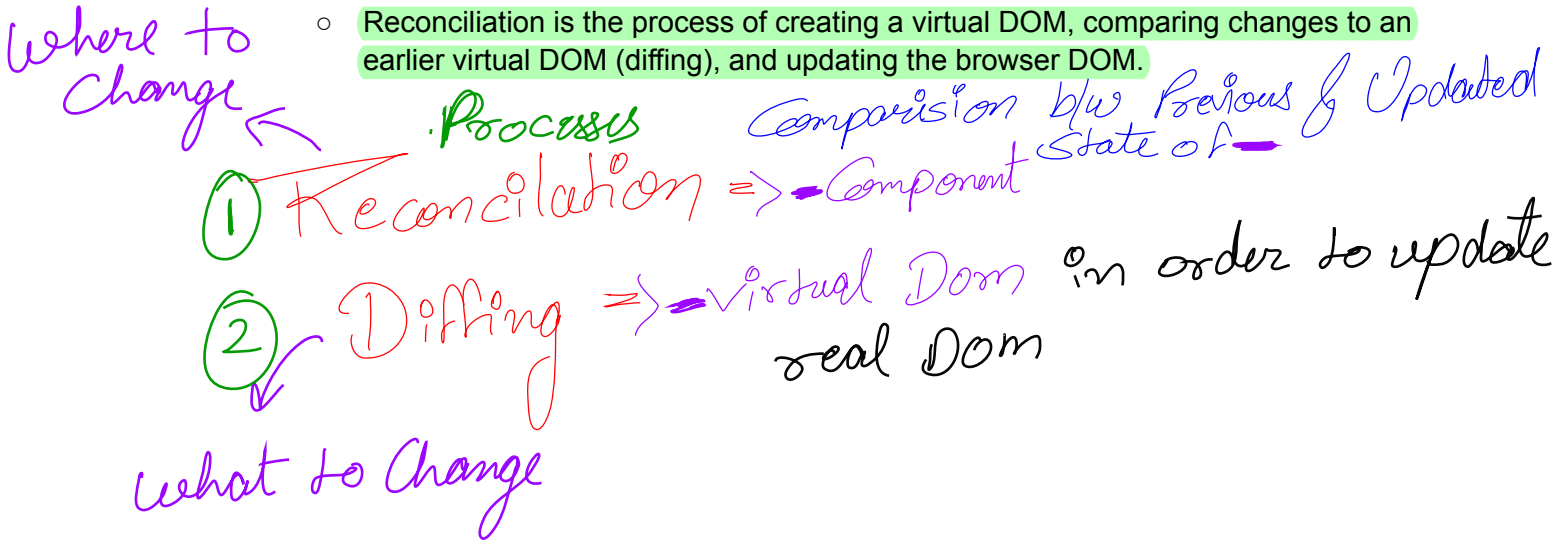
case the div container is the one we are returning, and we are unable to return any tags outside of it.

- Example:

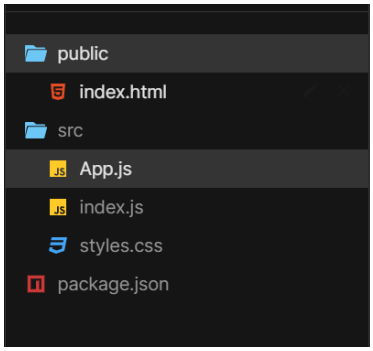
```
import React from 'react'
export default function Demo() {
  return (
    <div>
        <h1>Hello Everyone!</h1>
    </div>
  )
}
```

3. Component rendering:

- **Virtual DOM:**
  - The virtual DOM (VDOM) is a programming concept in which a library like ReactDOM maintains an ideal, or "virtual," representation of a user interface in memory and keeps it in sync with the "actual" DOM. This is referred to as *reconciliation*.
  - One of the reasons React is quick is because it keeps a replica of the Real DOM called the *Virtual DOM.* This is merely a virtual version of the DOM. Real DOM manipulations are expensive.
  - React produces a new virtual DOM and compares it to the old one whenever we make changes or add data. *Diffing* is the name of the comparison procedure; the modifications are then batch-processed and the real DOM is updated with the fewest changes possible without repainting the entire DOM.
  - Reconciliation is the process of creating a virtual DOM, comparing changes to an earlier virtual DOM (diffing), and updating the browser DOM.



- **Folder structure:**
  In public folder, the main file is index.html which contains a div with the id is equal root which is updated by index.js



public/index.html/

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
  <meta name="theme-color" content="#000000">
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
  <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
  <title>React App</title>
</head>

<body>
  <noscript>
    You need to enable JavaScript to run this app.
  </noscript>
  <div id="root"></div>
</body>
</html>
```

Index.html contains:

```
<div id="root"></div>
```

Which is responsible for UI change.

- Reconciliation :- The process of figuring out what part of a component need to be updated when its state changes.
- Diffing : Reconciliation algorithm or virtual DOM algorithm.

compares the previous & updated VDOM representation to determine the specific changes that need to be made in order to update real DOM.

index.js

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";

import App from "./App";

const rootElement = document.getElementById("root");
const root = createRoot(rootElement);

root.render(
 <StrictMode>
   <App />
 </StrictMode>
);
```

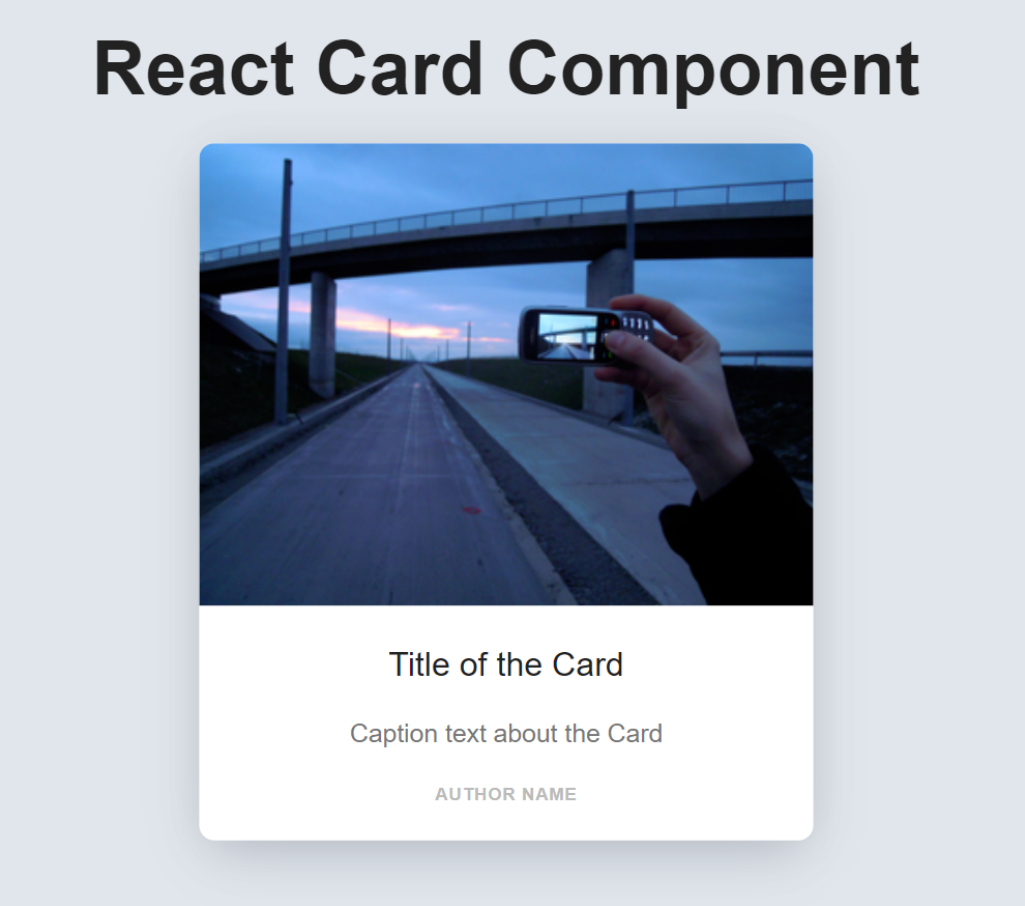
App.js is rendered here as a tag. First it is imported into the index.js and then it is rendered.

Let's make reusable components:
For this App.js

```
import "./styles.css";
export default function App() {
 return (
   <div className="App">
     <div className="header">
       <h1>React Card Component</h1>
     </div>
     <div className="cards">
       <Card />
     </div>
   </div>
 );
}
const Card = () => {
 return (
   <div className="card">
     <img

src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/b6/Image_created_with_a_mobile_phone.png/330px-Image_created_with_a_mobile_phone.png"
       alt="img"
     />
     <div className="card-body">
```

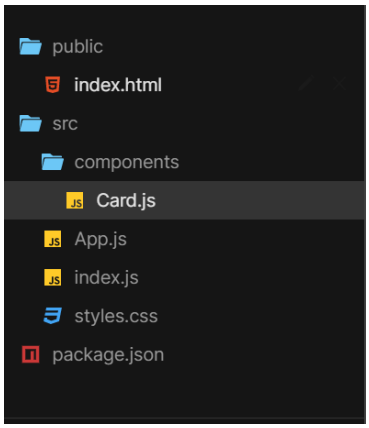
```
        <h2>Title of the Card</h2>
        <p>Caption text about the Card</p>
        <h5>Author name</h5>
      </div>
    </div>
  );
};
```

Whose output is :



Without changing the output we will modularize the code and change the structure of it.

We will create a folder called components and a file in it as Card.js



Inside the Card.js we will put the function which was there in App.js.

Card.js

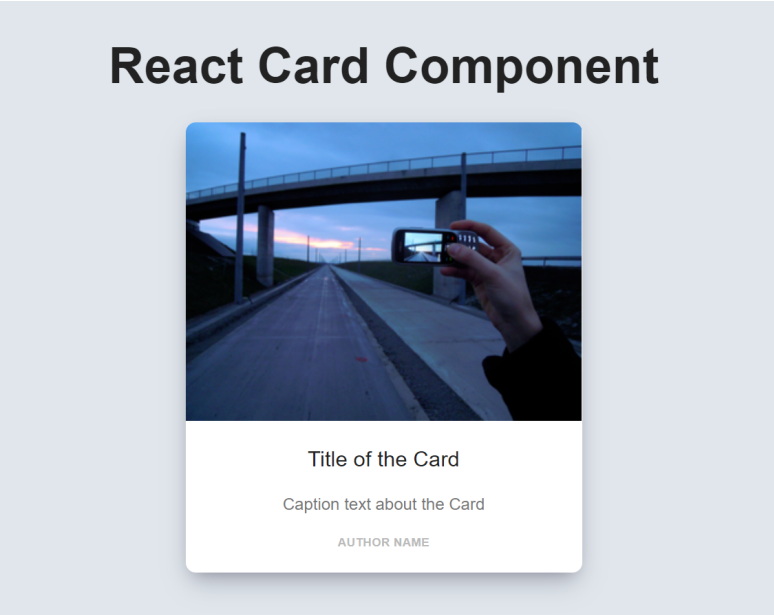
```
const Card = () => {
 return (
   <div className="card">
     <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/b/b6/Image_created_with_a_mobile_phone.png/330px-Image_created_with_a_mobile_phone.png"
       alt="img"
     />
     <div className="card-body">
       <h2>Title of the Card</h2>
       <p>Caption text about the Card</p>
       <h5>Author name</h5>
     </div>
   </div>
 );
};

export default Card;
```

And in App.js we have to import the Card component and render it as a tag.

App.js:

```
import "./styles.css";
import Card from "./components/Card";
export default function App() {
 return (
   <div className="App">
     <div className="header">
       <h1>React Card Component</h1>
     </div>
     <div className="cards">
       <Card />
     </div>
   </div>
 );
}
```

And here we are getting the same output after applying the appropriate styling to it.



Here