Styling the components and Complex Components

Styling the components

There are different ways to style the components in React.

- Inline CSS – using the style attribute
- CSS in JS - Using Libraries JSS and Styled Components
- CSS Modules - css loader and Sass & SCSS
- Stylable

Inline CSS in Detail

Consider the below code to provide red color to the contents of h1. But this is not a react way of providing the styling to the components.

```
render()
{
    return (<h1 style = {color:"red"}>welcome to styling to components</h1>)
}
```

We create objects of style and render it inside the components in style attribute using the React technique. Let us understand this through an example code.

```
<body>
    <div id = "root"></div>
    <script type = "text/babel">
        class Letter extends React.Component{
            render() {
 //Object letterstyle contains all key-value pairs of styling  to be applied to letter
                var letterstyle = {
                marginRight: "10px", // observe this comma
                textAlign: "center",backgroundColor:"blue", //observe the camelcase
                color:"olive",  padding: "20px", display:"inline"      }
                return <h1 style = {letterstyle}> {this.props.children}</h1>
            //Setting the style attribute to refer to the object
```

```
                    }
}
            ReactDOM.render(<div><Letter>A</Letter>
                    <Letter>E</Letter>
                    <Letter>I</Letter>
                    <Letter>O</Letter>
                    <Letter>U</Letter>    </div>
                    ,document.getElementById("root"))
        </script>
    <body>
```

If we execute the above code, All letters will have the same background color. To avoid this, background color must be sent suring the component rendering.

```
<script type = "text/babel">
        class Letter extends React.Component{
                render() {
 //Object letterstyle contains all key-value pairs of styling  to be applied to letter
                    var letterstyle = {
                    marginRight: "10px", // observe this comma
                    textAlign: "center", backgroundColor: this.props.bgcolor,
                    color:"olive",   padding: "20px"         }
                    return <h1 style = {letterstyle}> {this.props.children}</h1>
                    }
        }
        ReactDOM.render(<div><Letter bgcolor = "pink"> A</Letter>
                    <Letter bgcolor = "red">E</Letter>
                    <Letter bgcolor = "#0000FF">I</Letter>
                    <Letter bgcolor = "#BBEE00">O</Letter>
                    <Letter bgcolor = "#EE00BB">U</Letter>          </div>
                    ,document.getElementById("root"))
        </script>
```

Output of above code is as below:

Advantage of creation of style object is , it provides flexibility to add more styling properties as and when required.

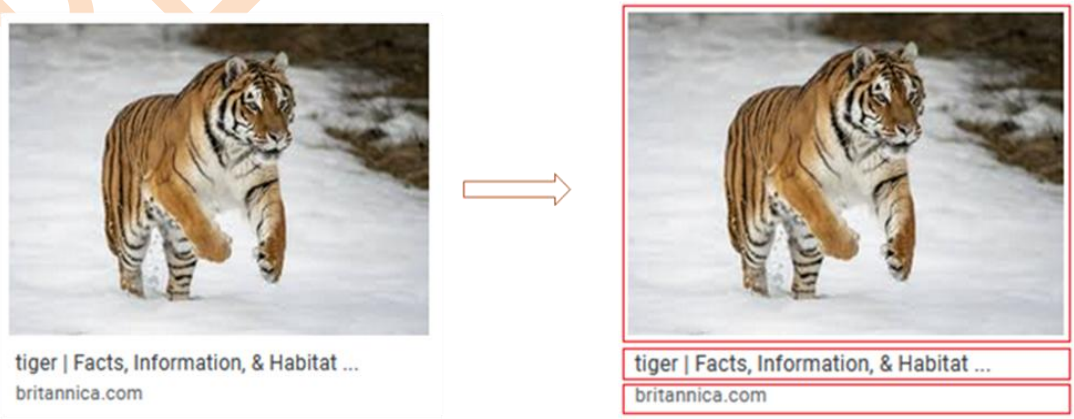
Complex Components

It is nothing but **building the component that uses other user defined components.** Also known as **component composition. React** lets you split the UI into smaller independent pieces so that you can reason about each piece in isolation. Using components rather than building the UI in a monolithic fashion also encourages reuse.. A component takes inputs (called properties) and its output is the rendered UI of the component. We will put together fine-grained components to build a larger UI.

Approach followed to build complex components:

- Identify the major visual elements
- Breaking them into individual components

Consider this example:



The Main complex component consists of **Image rendering, Caption rendering and Link rendering** on the web page.

The code for Main complex component is as below.

```
class Main extends React.Component{
        render() {
                return(<div>
                        <ReactImage />
                        <ReactCaption/>
                        <ReactLink/>
                </div>)
        }
}
```

Codes for ReactImage, ReactCaption and ReactLink is as below.

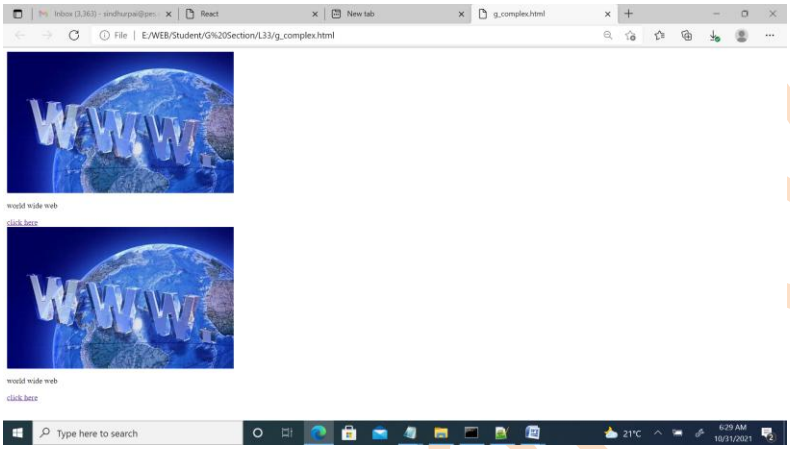
```
class ReactImage extends React.Component{
        render(){
                return (<img src = "www.jfif"/>)
        }

class ReactCaption extends React.Component{
        render(){
                return (<p>world wide web</p>)
        }
class ReactLink extends React.Component{
        render(){
                return (<a href = "https://www.google.com" >click here</a>)
        }
```

When we render this using ReactDOM.render,

ReactDOM.render(<**div**><**Main/**><**Main/**></**div**>**,** document.getElementById("root"))

, we get same image, caption and link both the times. This is because no attributes are sent during the component call. Also, clicking on the link, navigate to the same page as mentioned in the code.



Refer to the below code to have dynamic values for components. This adds this.props to access the values sent during the component call

```
class ReactImage extends React.Component{
        render()
        {       return (<img src = {this.props.src}/>)          }
}
class ReactCaption extends React.Component{
        render()
        {       return (<p>{this.props.caption}</p>)                    }
}
class ReactLink extends React.Component{
        render()
        {return (<a href = {this.props.href} >{this.props.link}</a>)          }
}
ReactDOM.render(<div>
        <Main src = "www.jfif" caption = "world" href =
```

"https://www.yahoo.com" link = "click here for yahoo"/>

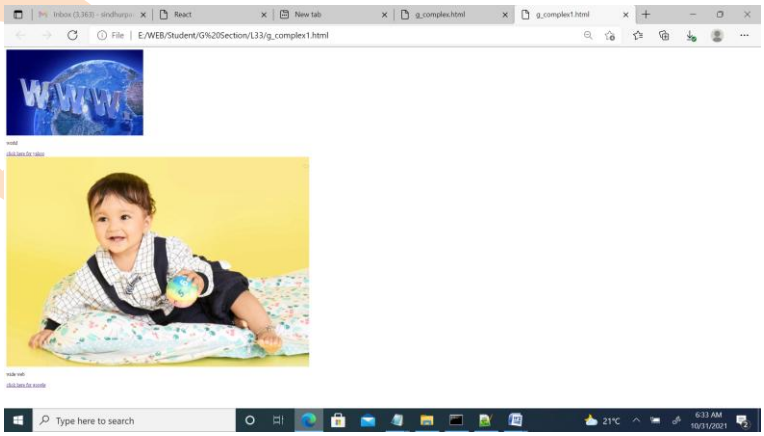
<Main src = "thaksh.jpg" caption = "wide web" href = "https://www.google.com" link = "click here for google"/> , document.getElementById("root"**))**

Transferring the properties:

If we keep the ReactDOM.render code same as previous, code doesn't throw any error. But no output as well. So think about where we have gone wrong. Observe that we have to **transfer the properties from Main component to three of the other components**. This is possible using the **spread operator(…)**

```
class Main extends React.Component{
        render(){    return(<div>
                     <ReactImage {...this.props}/>
                     <ReactCaption {...this.props}/>
                     <ReactLink {...this.props}/>
                </div>)  }
}
```

Output: Observe that rendering the same component with different values is possible using the attributes and props.



Think about having the same size for rendered images..!!