Map

The .map() method allows you to run a function on each item in the array, returning a new array as the result.

const Map = () =>{

    const array = [1, 2, 3, 4, 5];

    // this part is basic functionality map

    const arrayNumbers = array.map((number) => {

        return number \* 2;

    });

    console.log(arrayNumbers);

    console.log(array);

    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    //this returns the html for array objects

    // const arrayNumbersHtml = array.map((number) => {

    //     return <li>{number}</li>

    // });

    // return(

    //     <div>

    //         <h2>Example of React Map</h2>

    //         <ul>{arrayNumbersHtml}</ul>

    //     </div>

    // );

    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    //map within the return Html

    return (

        <div>

            <h2>Example three of React Map</h2>

            {array.map((item)=>{

            return (<li key={item.toString()}>

                        Index {item.toString()} - {item}

                    </li>)

            })}

        </div>

    );

    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

export default Map;

**Keys**

Keys help React identify which items have changed, are added, or removed. Keys should be given to the elements inside the array to give the elements a stable identity:

**React CSS**

CSS in React is used to style the React App or Component. The style attribute is the most used attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in camelCased properties rather than a CSS string.

1. Inline Styling
2. CSS Stylesheet
3. CSS Module
4. Styled Components
5. **Inline Styling**

The inline styles are specified with a JavaScript object in camelCase version of the style name. Its value is the style’s value, which we usually take in a string.

Ex: **<h1 style={{color: "Green"}}>Inline Styling!</h1>**

**camelCase Property Name**

If the properties have two names, like **background-color**, it must be written in camel case syntax.

Ex: **<p style={{backgroundColor: "lightgreen"}}>camelCase</ p>**

**Using JavaScript Object**

The inline styling also allows us to create an object with styling information and refer it in the style attribute.

Ex: const mystyle = {

color: "Green",

backgroundColor: "lightBlue",

padding: "10px",

fontFamily: "Arial"

};

**<h1 style={mystyle}>JavaScript Object</h1>**

1. **CSS Stylesheet**

You can write styling in a separate file for your React application, and save the file with a .css extension. Now, you can **import** this file in your application.

Ex:import "./styles.css";

export default function App() {

return (

<div className="teststyle-wrapper">

<div className="teststyle-quote">

<p>This is one of the best developer blogs on the planet! I read it daily to improve my skills</p>

</div>

</div>

);

}

CSS stylesheet(styles.css)

.teststyle-wrapper {

text-align: center;

max-width: 950px;

margin: 0 auto;

border: 1px solid #e6e6e6;

padding: 40px 25px;

margin-top: 50px;

}

.teststyle-quote p {

line-height: 1.5;

font-weight: 300;

margin-bottom: 25px;

font-size: 1.375rem;

}

1. **CSS Module**

* You can create CSS Module with the **.module.css** extension like a **myStyles.module.css** name.
* It is available only for the component which imports it, means any styling you add can never be applied to other components without your permission.
* You never need to worry about name conflicts.

**Ex:**

**import** styles from './myStyles.module.css';

**class** ExampleComp **extends** React.Component {

  render() {

**return** (

      <div>

      <h1 className={styles.mystyle}>Hello JavaTpoint</h1>

      <p className={styles.parastyle}>It provides great CS tutorials.</p>

      </div>

    );

  }

}

export **default** ExampleComp ;

**myStyles.module.css**

.mystyle {

  background-color: #cdc0b0;

  color: Red;

}

.parastyle{

  color: Green;

  font-family: Arial;

}

1. **Styled Components**

Styled-components is a library for React. It uses enhance CSS for styling React component systems in your application, which is written with a mixture of JavaScript and CSS.

The styled-components provides:

* Automatic critical CSS.
* No class name bugs.
* Easier deletion of CSS.
* Simple dynamic styling.
* Painless maintenance.
* Installation.

The styled-components library takes a single command to install in your React application. which is: $ npm install styled-components --save

Example:

Here, we create a variable by selecting a particular HTML element such as <div>, <Title>, and <paragraph> where we store our style attributes. Now we can use the name of our variable as a wrapper <Div></Div> kind of React component.

**import** React from 'react';

**import** ReactDOM from 'react-dom';

**import** styled from 'styled-components';

**class** App **extends** React.Component {

  render() {

**const** Div:any = styled.div`

            margin: 20px;

            border: 5px dashed green;

            &:hover {

            background-color: ${(props:any) => props.hoverColor};

            }

            `;

**const** Title = styled.h1`

            font-family: Arial;

            font-size: 35px;

            text-align: center;

            color: palevioletred;

            `;

**const** Paragraph = styled.p`

            font-size: 25px;

            text-align: center;

            background-Color: lightgreen;

            `;

**return** (

       <div>

            <Title>Styled Components Example</Title>

            <p></p>

            <Div hoverColor="Orange">

                 <Paragraph>Hello JavaTpoint!!</Paragraph>

            </Div>

        </div>

    );

  }

}

export **default** App;