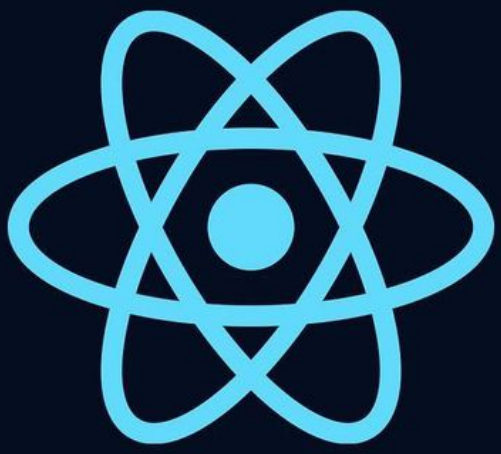
JS

React Cheatsheet (Part II)



Saad Irfan
@DevWithSaad

JS

Components and Hooks

Use functional components and hooks (like useState and useEffect) for a more concise and modular code structure.

index.js

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

const MyComponent = () => {
    const [count, setCount] = useState(0);

    useEffect(() => {
        // Effect logic ...
    }, [count]);

    return <div>{count}</div>;
}
```

Saad Irfan
@DevWithSaad

JS

Conditional Rendering

Render different components or elements based on conditions using the ternary operator.

index.js

```
const MyComponent = ({ isLoggedIn }) => {
    return (
        <>
            <Nav />
            {isLoggedIn ? <UserDashboard /> : <LoginScreen />}
        </>
    );
};
```

Saad Irfan
@DevWithSaad

JS

Controlled Components

Use controlled components to handle user input and maintain the input's state within the component.

index.js

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

const InputForm = () => {
  const [value, setValue] = useState('');

  const handleChange = (event) => {
    setValue(event.target.value);
  };

  return (
    <input type="text" value={value} onChange={handleChange} />
  );
};
```

Saad Irfan
@DevWithSaad

JS

Mapping Data

Map an array of data to JSX elements to dynamically render a list of items.

index.js

```
const MyList = ({ items }) => {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
};
```

Saad Irfan
@DevWithSaad

JS

Using Fragments

Use React Fragments to avoid adding unnecessary wrapping elements when rendering multiple elements.

index.js

```
const MyComponent = () => {
  return (
    <>
      <h1>Title</h1>
      <p>Paragraph 1</p>
      <p>Paragraph 2</p>
    </>
  );
};
```

Saad Irfan
@DevWithSaad