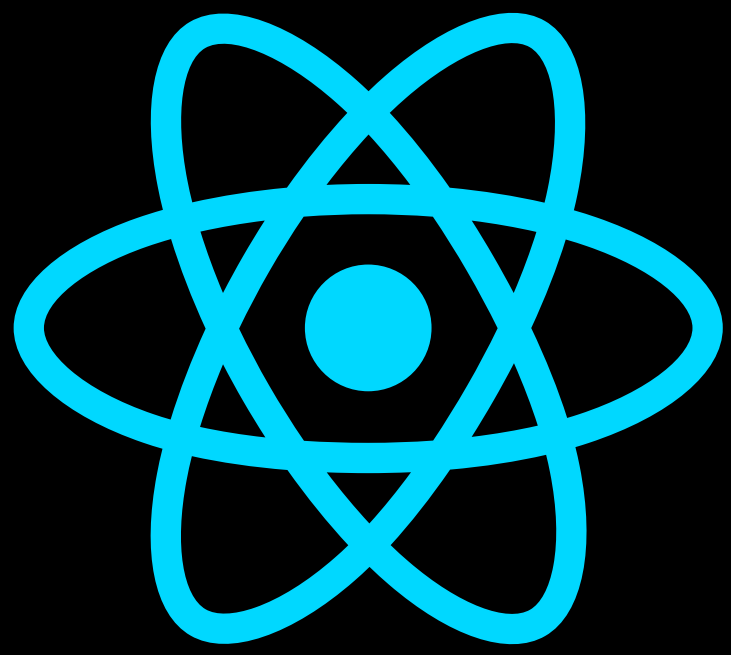

Quick Reference

Component Creation:

Define reusable UI elements using functions or classes.

```jsx
import React from 'react';

const MyComponent = () => {
    return <div>Hello, React!</div>;
};
```

Props:

Pass data from parent to child components to customize behavior.

```
const Greeting = (props) => {
    return <div>Hello, {props.name}!</div>;
};

// Usage
<Greeting name="World" />;
```

State:

Manage component-specific data that can change over time.

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

const Counter = () => {
    const [count, setCount] = useState(0);
    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={
                () => setCount(count + 1)}>Increment</button>
        </div>
    );
};
```

Event Handling:

Respond to user interactions like clicks or input changes

```jsx
const Button = () => {
    const handleClick = () => {
        alert('Button clicked!');
    };
    return <button onClick={handleClick}>Click Me</button>;
};
```

Conditional Rendering:

Display different content based on conditions or variables.

```jsx
const ConditionalComponent = ({ condition }) => {
    return condition ? <div>Condition is true</div> :
     <div>Condition is false</div>;
};
```

Lists and Keys:

Render dynamic lists of items efficiently, with unique identifiers.

```jsx
const List = ({ items }) => {
    return (
        <ul>
            {items.map(item =>
            <li key={item.id}>{item.name}</li>)}
        </ul>
    );
};
```

Effect Hook:

Call useEffect at the top level of your component to declare an Effect

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

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

    useEffect(() => { document.title = `Clicked ${count} times`; }, [count]);

    return (
        <div>
            <p>Clicked {count} times</p>
            <button onClick={() => setCount(count + 1)}>Click me</button>
        </div>
    );
};

export default MyComponent;
```