Hello World

```
// Render a div with Hello World in the body
import React from "react";
import ReactDOM from "react-dom";

ReactDOM.render(
  <div>Hello World</div>
 , document.getElementsByTagName("body")[0]);
```

JSX Conditional Rendering

Ternary Expression

```
<div>
  {isThisTrue
  ? <TrueComponent />
  : <FalseComponent />}
</div>
```

Shorthand

```
<div>
  {isThisTrue && (
     <TrueComponent />
  )}
  {!isThisTrue && (
     <FalseComponent />
  )}
</div>
```

JSX Loops

```
let data = [1,2,3,4,5];

return <div>
  {data.map(item =>
    <span>item: {item} </span>
  )}
</div>
```

ES6 Component

```
// Use ES6 to create a reusable component that can
// be rendered by using <Hello /> in JSX
import React from "react";
import ReactDOM from "react-dom";

class Hello extends React.Component {
  render() {
    return (
       <div>Hello World</div>
    )
  }
}

ReactDOM.render(<Hello />, document.getElementsByTagName("body")[0]);
```

ES6 Component using Properties

```
// This component receives properties when invoked and
// uses those properties in the render method
import React from "react";
import ReactDOM from "react-dom";

class Hello extends React.Component {
  render() {
    return (
      <div>
        Hello { this.props.name }
      </div>
    )
  }
}

ReactDOM.render(<Hello name="World"/>,
       document.getElementsByTagName("body")[0]);
```

Component Lifecycle

```
class Lifecycles extends React.Component {
  constructor (props) {
    // Mounting
    // Happens before rendering
  }

  componentWillMount() {
    // Mounting
    // Deprecated, you shouldn't use this
  }

  componentWillReceiveProps (newProps) {
    // Updating
    // When new properties arrive, use setState() here
  }

  shouldComponentUpdate (newProps, newState) {
    // Updating
    // Cancels the rendering if returns false
  }

  componentWillUpdate (newProps, newState) {
    // Updating
    // Before component update
  }

  render() {
    // The rendering function builds the DOM to be rendered
  }

  componentDidUpdate (prevProps, prevState){
    // Updating
    // Component just updated with new state
  }

  componentDidMount() {
    // Mounting
    // After the rendering is done
  }

  componentWillUnmount() {
    // Mounting
    // Before unmounting a component
  }

  componentDidCatch() {
    // Mounting
    // Used to catch errors
  }
}
```

Events and Simple State Management

```
// This component has a state and has a method to
// change the state of the component
class Counter extends React.Component {
  state = {
    counter: 0
  };

  constructor(props) {
    super(props);
    // This is so that we can use this.setState in the
    // incrementCounter method.  Without it, this is undefined
    this.incrementCounter = this.incrementCounter.bind(this);
  }

  incrementCounter() {
    // This method updates the state of the component and
    // increases the counter by one.
    this.setState({counter: this.state.counter + 1});
  }

  render() {
    return (
      <div>
        Counter: {this.state.counter}
        <button onClick={this.incrementCounter}>+1</button>
      </div>
    )
  }
}
```

About the Author

Joel Lord is passionate about web and technology in general. He likes to learn new things but most of all, he likes to share his discoveries. He does so by travelling to various conferences all across the globe. He graduated from college in computer programming in the last millennium. Apart from a little break to get his BSc in computational astrophysics, he was always in the industry. As a technical evangelist with Auth0, he meets with developers to help them make the web a safer place.

iJS International JavaScript Conference

React Cheat Sheet

Concept & Content by Joel Lord

Rendering Children Nodes

```
class Messages extends React.Component {
  render() {
    // This will display everything between the opening
    // and closing Messages tags
    return (
      <div>
        {this.props.children}
      </div>
    )
  }
}

class Main extends React.Component {
  render() {
    return (
      <Messages>
        <div>Message 1</div>
        <div>Message 2</div>
      </Messages>
    )
  }
}
```

Basic Form with Input Box

```
class BasicForm extends React.Component {
  state = {
    inputValue: ""
  };

  constructor(props) {
    super(props);
    this.inputChanged = this.inputChanged.bind(this);
  }

  inputChanged(event) {
    // When the user types in the input box, we need to
    // update the state so that the re-render is done with
    // the new value
    this.setState({ inputValue: event.target.value })
  }

  render() {
    return (
      <div>
        Input box says: { this.state.inputValue } <br/>
        <input onChange={ this.inputChanged } value={ this.state.inputValue } type="text"/>
      </div>
    )
  }
}
```

Routing

Define the Routes

```
import React from 'react';
import ReactDOM from "react-dom";
import Main from "./components/Main";
import Secret from "./components/Secret";
import NotFound from "./components/NotFound";
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";

class App extends React.Component {
  render() {
    return (
      <Router>
        <Switch>
          <Route exact path="/" render={(props) => <Main {...this.props} />} />
          <Route path="/secret" component={Secret} />
          <Route component={NotFound} />
        </Switch>
      </Router>
    );
  }
}
```

Linking to a Route

```
import { Link } from "react-router-dom";

class NavBar extends React.Component {
  render() {

    return (
      <Link to="/">Home</Link>
    );
  }
}
```

Redirecting to a Route

```
<Redirect to="/" push={true} />
```

Typechecking

Basic

```
BasicComponent.propTypes = {
  strings: PropTypes.string,
  numbers: PropTypes.number,
  functions: PropTypes.func,
  booleans: PropTypes.bool,
  required: PropTypes.any.isRequired
}
```

Arrays, Enums and Others

```
BasicComponent.propTypes = {
  cardinalPoints: PropTypes.oneOf([ "north", "south", "east", "west" ]),
  coordinates: PropTypes.arrayOf(PropTypes.number),
  person: PropTypes.shape({
    name: PropTypes.string,
    address: PropTypes.string
  })
}
```

React Cheat Sheet

Concept & Content by Joel Lord

Presented by: Auth0 entwickler.de International JavaScript Conference