



## REACT EXAMPLE CODE

```

import React, { Component }
  from "react";
import "../App.css";
import sendIcon from
  "../images/envelope.png";
import Button from
  "../components/Button";

class App extends Component {
  // Define starting state
  state = {
    message: "",
    chatLog: []
  };

  // Form elements need state
  // modifying methods
  onMsgChange (ev) {
    const {value} = ev.target;
    this.setState({
      message: value,
    });
  }

  // Lifecycle methods
  // First rendered to screen
  componentDidMount() {}
  // After props or state change
  componentDidUpdate() {}
  // Can prevent DOM update
  shouldComponentUpdate() {}

  render() {
    // Special method: Return JSX
    // that is to be rendered
    let {chatLog} = this.state;
    let count = chatLog.length;

    return (
      <div className="App">
        <h1>{count} messages</h1>
        {this.state.chatLog.map(
          text => (
            <p>{text}</p>
          )
        )}
        { /* Needs ".bind(this)"
           to keep context */ }
        <input
          value={this.state.message}
          onChange={this.onMsgChange
            .bind(this)} />
        <Button onClick={
          this.sendMsg.bind(this)}>
          <img src={sendIcon} />
          Send message
        </Button>
      </div>
    );
  }
}

```

## DEFINING COMPONENTS

```

// Class based component
class Button extends Component {
  render() {
    <button className="Button"
      onClick={this.props.onClick}>
      {this.props.children}
    </button>
  }
}

// Functional component
const Button = (props) => (
  <button className="Button"
    onClick={props.onClick}>
    {props.children}
  </button>
);
export default Button;

```

## USEFUL REACT PATTERNS

## Conditional rendering:

```

render() {
  if (!this.props.text) {
    return "Empty...";
  }
  // full render here ...
}

```

## Using map to loop through data:

```

<div>{
  data.map((item, i) => (
    <p onClick={this.handleClick
      .bind(this, i)}>
      {i}: {item}
    </p>
  ))
}</div>

```

## Using ternary operator:

```

<div>{
  this.props.image ? (
    <img src={this.props.image} />
  ) : <em>No image provided.</em>
}</div>

```

## Using ref to incorporate legacy JS:

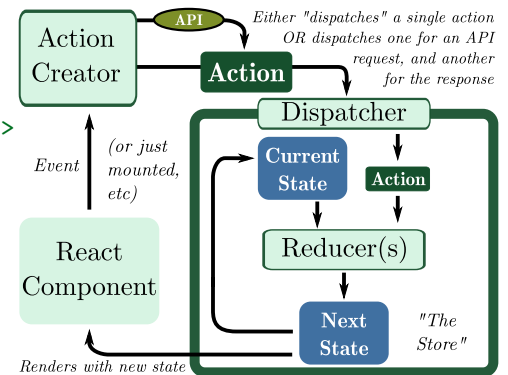
```

// Somewhere in JSX (e.g. render)
<div ref={el => { this.btn = el; }}>
  Click me!</div>
// Somewhere in JS (e.g. a method)
$(this.btn).modal();

```

## REACT, ROUTER, REDUX

## REACT REDUX



## Action Creators (found in actions/)

```

const doIncrement = () =>
  ({type: INCREMENT});
const addTodo = (item) =>
  ({type: ADD_TODO, text: item});

```

## Dispatching (found in components/)

```

let action =
  addTodo(this.state.text);
this.props.dispatch(action);

```

## Reducers (found in reducers/)

```

const initialState = {
  count: 0,
  todoList: [],
};
const todo = (state, action) => {
  switch (action.type) {
    case INCREMENT:
      return Object.assign({}, state, {
        count: state.count + 1,
      });
    case ADD_TODO:
      return Object.assign({}, state, {
        todoList: [...todoList,
          action.text],
      }); /* etc ... */
  }
}

```

## REACT ROUTER

```

<nav>
  <Link to="/about/">About</Link>
  <Link to="/post/"+postId+"/">
    Read More...</Link>
</nav>
<main>
  <Switch>
    <Route path="/about/"
      component={About} />
    <Route path="/post/:id/"
      component={BlogPost} />
  </Switch>
</main>

```