React

A JavaScript library for building user interfaces

Get Started Take the Tutorial

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Bas

Build encapsulated
compose them to m

Since component lo
you can easily pass
the DOM.

A Simple Component

React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX. Input data that is passed into the component can be accessed by `render()` via `this.props`.

JSX is optional and not required to use React. Try the Babel REPL to see the raw JavaScript code produced by the JSX compilation step.

LIVE JSX EDITOR ☑ JSX?

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

root.render(<HelloMessage name="Taylor" />);
```

RESULT

Hello Taylor

A Stateful Component

In addition to taking input data (accessed via `this.props`), a component can maintain internal state data (accessed via `this.state`). When a component's state data changes, the rendered markup will be updated by re-invoking `render()`.

LIVE JSX EDITOR ☑ JSX?

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  tick() {
    this.setState(state => ({
      seconds: state.seconds + 1
    }));
  }

  componentDidMount() {
    this.interval = setInterval(() => this.tick(), 1000);
  }
```

RESULT

Seconds: 24

An Application

Using `props` and `state`, we can put together a small Todo application. This example uses `state` to track the current list of items as well as the text that the user has entered. Although event handlers appear to be rendered inline, they will be collected and implemented using event delegation.

LIVE JSX EDITOR ☑ **JSX?**

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { items: [], text: '' };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  render() {
    return (
      <div>
        <h3>TODO</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <label htmlFor="new-todo">
            What needs to be done?
```

RESULT

TODO

What needs to be done?

Add #1

A Component Using External Plugins

React allows you to interface with other libraries and frameworks. This example uses **remarkable**, an external Markdown library, to convert the `<textarea>`'s value in real time.

LIVE JSX EDITOR ✓ JSX?

```
class MarkdownEditor extends React.Component {
  constructor(props) {
    super(props);
    this.md = new Remarkable();
    this.handleChange = this.handleChange.bind(this);
    this.state = { value: 'Hello, **world**!' };
  }

  handleChange(e) {
    this.setState({ value: e.target.value });
  }

  getRawMarkup() {
    return { __html: this.md.render(this.state.value) };
  }
```

RESULT

Input

Enter some markdown

```
Hello, **world**!
```

Output

Hello, **world**!

Get Started Take the Tutorial