

# 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-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

## Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using [React Native](#).

## 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?	RESULT
<pre>class HelloMessage extends React.Component {   render() {     return (       &lt;div&gt;         Hello {this.props.name}       &lt;/div&gt;     );   } }  ReactDOM.render(   &lt;HelloMessage name="Taylor" /&gt;,   document.getElementById('hello-example') );</pre>	<input checked="" type="checkbox"/>	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?	RESULT
<pre>class Timer extends React.Component {   constructor(props) {     super(props);     this.state = { seconds: 0 };   }    tick() {     this.setState(state =&gt; ({       seconds: state.seconds + 1     }));   }    componentDidMount() {     this.interval = setInterval(() =&gt; this.tick(), 1000);   } }</pre>	<input checked="" type="checkbox"/>	Seconds: 4

## 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?	RESULT
<pre>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 (       &lt;div&gt;         &lt;h3&gt;TODO:&lt;/h3&gt;         &lt;TodoList items={this.state.items} /&gt;         &lt;form onSubmit={this.handleSubmit}&gt;           &lt;input type="text" value={this.state.text} /&gt;           &lt;button type="submit"&gt;Add #1&lt;/button&gt;         &lt;/form&gt;       &lt;/div&gt;     );   } }</pre>	<input checked="" type="checkbox"/>	<b>TODO</b> What needs to be done? <input type="text"/> <input type="button" value="Add #1"/>

## A Component Using External Plugins

LIVE JSX EDITOR	JSX?	RESULT
<pre>class MarkdownEditor extends React.Component {   constructor(props) {     super(props);     this.state = { text: '' };   }    render() {     return (       &lt;div&gt;         &lt;input type="text" value={this.state.text} /&gt;         &lt;button type="button" value="Save" /&gt;       &lt;/div&gt;     );   } }</pre>	<input checked="" type="checkbox"/>	Input

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.

```

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) };
}
```

Enter some markdown

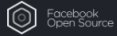
Hello, \*\*world\*\*!

## Output

Hello, **world**!

Get Started

Take the Tutorial >



Copyright © 2021 Facebook Inc.

### DOCS

Installation

Main Concepts

Advanced Guides

API Reference

Hooks

Testing

Concurrent Mode (Experimental)

Contributing

FAQ

### COMMUNITY

Code of Conduct

Community Resources

### CHANNELS

GitHub

Stack Overflow

Discussion Forums

Reactiflux Chat

DEV Community

Facebook

Twitter

### MORE

Tutorial

Blog

Acknowledgements

React Native

Privacy

Terms