



React

用于构建用户界面的
JavaScript 库

快速开始 [入门教程 >](#)

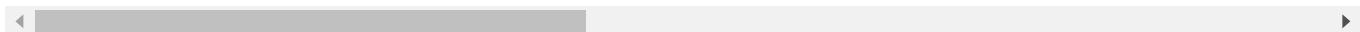
声明式

React 使创建交互式 UI 变得轻而易举。为你应用的每一个状态设计简洁的视图，当数据改变时 React 能有效地更新并正确地渲染组件。

以声明式编写 UI，可以让你的代码更加可靠，且方便调试。

组件化

创建拥有各自状态的组件逻辑使用 JavaScript 递数据，并使得状态



简单组件

React 组件使用一个名为 `render()` 的方法，接收输入的数据并返回需要展示的内容。在示例中这种类似 XML 的写法被称为 JSX。被传入的数据可在组件中通过 `this.props` 在 `render()` 访问。

使用 React 的时候也可以不使用 JSX 语法。尝试使用 [Babel REPL](#)，了解 JSX 被编译成原生 JavaScript 代码的步骤。

LIVE JSX EDITOR

JSX?

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
```

 文档 教程 博客 社区

```
    );
}
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

RESULT

Hello Taylor

有状态组件

除了使用外部数据（通过 `this.props` 访问）以外，组件还可以维护其内部的状态数据（通过 `this.state` 访问）。当组件的状态数据改变时，组件会再次调用 `render()` 方法重新渲染对应的标记。

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: 7



应用

使用 `props` 和 `state`，我们可以创建一个简易的 Todo 应用。在示例中，我们使用 `state` 来保存现有的待办事项列表及用户的输入。尽管事件处理器看似被内联地渲染，但它们其实只会被事件委托进行收集和调用。

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?
          </label>
          <input type="text" value={this.state.text} onChange={this.handleChange} />
          <button type="submit">Add #1</button>
        </form>
      </div>
    );
  }
}
```

RESULT

TODO

What needs to be done?

在组件中使用外部插件

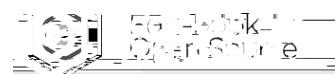
React 允许你结合其他框架或库一起使用。示例中使用了一个名为 **remarkable** 的外部 Markdown 库。它可以实时转换 `<textarea>` 里的内容。

[文档](#) [教程](#) [博客](#) [社区](#)[贡献](#)[FAQ](#)[Facebook](#)

社区

[社区资源](#)[工具](#)

其他

[教程](#)[博客](#)[致谢](#)[React Native](#)

Copyright © 2019 Facebook Inc.

[印记中文](#)