var Component = React.createClass({

render: function () {

return <div>Hello {this.props.name}</div>;

}

});

ReactDOM.render(<Component name="John" />, document.body);

*Nesting*

Nest components to separate concerns. See [multiple components](http://facebook.github.io/react/docs/multiple-components.html).

var UserAvatar = React.createClass({...});

var UserProfile = React.createClass({...});

var Info = React.createClass({

render() {

return <div>

<UserAvatar src={this.props.avatar} />

<UserProfile username={this.props.username} />

</div>;

}

});

*States & Properties*

Use [props](https://facebook.github.io/react/docs/tutorial.html#using-props) (this.props) to access parameters passed from the parent. Use [states](https://facebook.github.io/react/docs/tutorial.html#reactive-state)(this.state) to manage dynamic data.

<MyComponent fullscreen={true} />

// props

this.props.fullscreen //=> true

// state

this.setState({ username: 'rstacruz' });

this.replaceState({ ... });

this.state.username //=> 'rstacruz'

render: function () {

return <div className={this.props.fullscreen ? 'full' : ''}>

Welcome, {this.state.username}

</div>;

}

**Setting defaults**

Pre-populates this.state.comments and this.props.name.

React.createClass({

getInitialState: function () {

return { comments: [] };

},

getDefaultProps: function () {

return { name: "Hello" };

}

);

*Component API*

These are methods available for Component instances. See [Component API](http://facebook.github.io/react/docs/component-api.html).

ReactDOM.findDOMNode(c) // 0.14+

React.findDOMNode(c) // 0.13

c.getDOMNode() // 0.12 below

c.forceUpdate()

c.isMounted()

c.state

c.props

c.setState({ ... })

c.replaceState({ ... })

c.setProps({ ... }) // for deprecation

c.replaceProps({ ... }) // for deprecation

c.refs

**Component specs**

Methods and properties you can override. See [component specs](http://facebook.github.io/react/docs/component-specs.html).

|  |  |
| --- | --- |
| [render()](http://facebook.github.io/react/docs/component-specs.html#render) |  |
| [getInitialState()](http://facebook.github.io/react/docs/component-specs.html#getinitialstate) |  |
| [getDefaultProps()](http://facebook.github.io/react/docs/component-specs.html#getdefaultprops) |  |
| [mixins: [ ... ]](http://facebook.github.io/react/docs/component-specs.html#mixins) | Mixins … [more](http://ricostacruz.com/cheatsheets/react.html#mixins) |
| [propTypes: { ... }](http://facebook.github.io/react/docs/component-specs.html#proptypes) | Validation … [more](http://ricostacruz.com/cheatsheets/react.html#property-validation) |
| [statics: { ... }](http://facebook.github.io/react/docs/component-specs.html#statics) | Static methods |
| [displayName: "..."](http://facebook.github.io/react/docs/component-specs.html#displayname) | Automatically filled by JSX |

*Lifecycle*

**Mounting**

Before initial rendering occurs. Add your DOM stuff on didMount (events, timers, etc). See [reference](http://facebook.github.io/react/docs/component-specs.html#mounting-componentwillmount).

|  |  |
| --- | --- |
| componentWillMount() | Before rendering (no DOM yet) |
| componentDidMount() | After rendering |

**Updating**

Called when parents change properties and .setState(). These are not called for initial renders. See [reference](http://facebook.github.io/react/docs/component-specs.html#updating-componentwillreceiveprops).

|  |  |
| --- | --- |
| componentWillReceiveProps*(newProps={})* | Use setState() here |
| shouldComponentUpdate*(newProps={}, newState={})* | Skips render() if returns false |
| componentWillUpdate*(newProps={}, newState={})* | Can’t use setState() here |
| componentDidUpdate*(prevProps={}, prevState={})* | Operate on the DOM here |

**Unmounting**

Clear your DOM stuff here (probably done on didMount). See [reference](http://facebook.github.io/react/docs/component-specs.html#unmounting-componentwillunmount).

|  |  |
| --- | --- |
| componentWillUnmount() | Invoked before DOM removal |

**Example: loading data**

See [initial AJAX data](http://facebook.github.io/react/tips/initial-ajax.html).

React.createClass({

componentWillMount: function () {

$.get(this.props.url, function (data) {

this.setState(data);

}.bind(this));

},

render: function () {

return <CommentList data={this.state.data} />

}

});

*DOM nodes*

**References**

Allows access to DOM nodes. See [References](http://facebook.github.io/react/docs/more-about-refs.html).

<input ref="myInput">

this.refs.myInput

ReactDOM.findDOMNode(this.refs.myInput).focus()

ReactDOM.findDOMNode(this.refs.myInput).value

**DOM Events**

Add attributes like onChange. See [events](https://facebook.github.io/react/docs/events.html).

<input type="text"

value={this.state.value}

onChange={this.handleChange} />

handleChange: function(event) {

this.setState({ value: event.target.value });

}

**Two-way binding**

Use [LinkedStateMixin](http://facebook.github.io/react/docs/two-way-binding-helpers.html) for easier two-way binding.

Email: <input type="text" valueLink={this.linkState('email')} />

React.createClass({

mixins: [React.addons.LinkedStateMixin]

});

this.state.email

*Property validation*

**Basic types**

Primitive types: .string, .number, .func, and .bool. See [propTypes](http://facebook.github.io/react/docs/reusable-components.html#prop-validation).

React.createClass({

propTypes: {

email: React.PropTypes.string,

seats: React.PropTypes.number,

settings: React.PropTypes.object,

callback: React.PropTypes.func,

isClosed: React.PropTypes.bool,

any: React.PropTypes.any,

}

});

**Required types**

Add .isRequired.

propTypes: {

requiredFunc: React.PropTypes.func.isRequired,

requiredAny: React.PropTypes.any.isRequired,

**React elements**

Use .element, .node.

propTypes: {

element: React.PropTypes.element, // react element

node: React.PropTypes.node, // num, string, element

// ...or array of those

**Enumerables**

Use .oneOf, .oneOfType.

propTypes: {

enum: React.PropTypes.oneOf(['M','F']), // enum

union: React.PropTypes.oneOfType([ // any

React.PropTypes.string,

React.PropTypes.number ]),

**Arrays and objects**

Use .array[Of], .object[Of], .instanceOf, .shape.

propTypes: {

array: React.PropTypes.array,

arrayOf: React.PropTypes.arrayOf(React.PropTypes.number),

object: React.PropTypes.object,

objectOf: React.PropTypes.objectOf(React.PropTypes.number),

message: React.PropTypes.instanceOf(Message),

object2: React.PropTypes.shape({

color: React.PropTypes.string,

size: React.PropTypes.number

}),

**Custom validation**

Supply your own function.

propTypes: {

customProp: function(props, propName, componentName) {

if (!/matchme/.test(props[propName])) {

return new Error('Validation failed!');

}

}

}

*Other features*

**Class set**

Manipulate DOM classes with [classnames](https://www.npmjs.org/package/classnames), previously known asReact.addons.classSet. See [Class set](http://facebook.github.io/react/docs/class-name-manipulation.html).

var cx = require('classnames');

render: function() {

var classes = cx({

'message': true,

'message-important': this.props.isImportant,

'message-read': this.props.isRead

});

return <div className={classes}>Great Scott!</div>;

}

**Propagating properties**

See [Transferring props](http://facebook.github.io/react/docs/transferring-props.html).

<VideoPlayer src="video.mp4" />

var VideoPlayer = React.createClass({

render: function() {

/\* propagates src="..." down to this sub component \*/

return <VideoEmbed {...this.props} controls='false' />;

}

});

**Mixins**

See [addons](https://facebook.github.io/react/docs/addons.html) for some built-in mixins.

var SetIntervalMixin = {

componentWillMount: function() { .. }

}

var TickTock = React.createClass({

mixins: [SetIntervalMixin]

}

[*Top level API*](https://facebook.github.io/react/docs/top-level-api.html)

React.createClass({ ... })

React.isValidElement(c)

ReactDOM.findDOMNode(c) // 0.14+

ReactDOM.render(<Component />, domnode, [callback]) // 0.14+

ReactDOM.unmountComponentAtNode(domnode) // 0.14+

ReactDOMServer.renderToString(<Component />) // 0.14+

ReactDOMServer.renderToStaticMarkup(<Component />) // 0.14+

*JSX patterns*

**Style shorthand**

See [inline styles](https://facebook.github.io/react/tips/inline-styles.html).

var style = { backgroundImage: 'url(x.jpg)', height: 10 };

return <div style={style}></div>;

**InnerHTML**

See [dangerously set innerHTML](https://facebook.github.io/react/tips/dangerously-set-inner-html.html).

function markdownify() { return "<p>...</p>"; }

<div dangerouslySetInnerHTML={{\_\_html: markdownify()}} />

**Lists**

var TodoList = React.createClass({

render: function() {

function item(itemText) {

return <li>{itemText}</li>;

};

return <ul>{this.props.items.map(item)}</ul>;

}

});