Basic class

```
React.createClass({
  render: function () {
    return (
      <div>Hello {this.props.name}</div>
    );
  }
});
```

Using components

```
var Component = React.createClass({ ... });
var compo = React.render(<Component />, mountnode);
```

States

```
this.setState({ editing: true });
this.state.editing === true
this.replaceState({ ... });
```

Properties

```
this.setProps({ fullscreen: true });
this.props.fullscreen === true
this.replaceProps({ ... });
```

API

```
c.getDOMNode() // deprecated 0.13
React.findDOMNode(c) // 0.13+
c.forceUpdate()
c.isMounted()
```

Methods

```
{
  render: function()
  getInitialState: function()
  getDefaultProps: function()
  mixins: []
  propTypes: {} / for validation /
  statics: { .. } / static methods /
  displayName: '..' / automatically filled in by jsx/
}
```

Lifecycle

```
componentWillMount()
componentDidMount()
// not on initial render
componentWillReceiveProps(props)
shouldComponentUpdate(props, state)
componentWillUpdate(props, state)
componentDidUpdate(prevProps, prevState)
componentWillUnmount()
// ...there is no DidUnmount or ReceiveState.
```

Initial States

```
React.createClass({
  getInitialState: function () {
    return {data: []};
  },
  render: function () {
    return (
      <CommentList data={this.state.data} />
    );
  }
});
```

Default Properties

```
React.createClass({
  getDefaultProps: function () {
    return {name: ''};
  }
);
```

Before Rendering

```
React.createClass({
  componentWillMount: function () {
    $.get(this.props.url, function (data) {
      this.setState(data);
    });
  },
  render: function () {
    return (
      <CommentList data={this.state.data} />
    );
  }
});
```

Actions

```
<form onSubmit={this.handleSubmit}>
  Name: <input ref="name">
</form>
React.createClass({
  handleSubmit: function (event) {
    name = this.refs['name'].getDOMNode().value;
    // see two-way binding below
  }
})
```

Two-way binding

```
React.createClass({
  mixins: [React.addons.LinkedStateMixin],
  getInitialState: function() {
    return {value: 'Hello!'};
  },
  render: function() {
    return <input type="text" valueLink={this.linkState('value')} />;
  }
});
// LinkedStateMixin adds a method to your React component called
// linkState().
```

Lists

```
var TodoList = React.createClass({
  render: function() {
    var createItem = function(itemText) {
      return <li>{itemText}</li>;
    };
    return <ul>{this.props.items.map(createItem)}</ul>;
  }
});
```

Property validations

```
React.createClass({
  propTypes: {
    // required
    requiredFunc: React.PropTypes.func.isRequired,
    requiredAny: React.PropTypes.any.isRequired,
    // primitives, optional by default
    bool: React.PropTypes.bool,
    func: React.PropTypes.func,
```

Property validations (cont)

```
    number: React.PropTypes.number,
    string: React.PropTypes.string,
  }
});
```

Class set

```
render: function() {
  var cx = React.addons.classSet;
  var classes = cx({
    'message': true,
    'message-important': this.props.isImportant,
    'message-read': this.props.isRead
  });
  // same final string, but much cleaner
  return <div className={classes}>Great Scott!</div>;
}
```

Propagating properties to children

```
var VideoPlayer = React.createClass({
  render: function() {
    return <VideoEmbed {...this.props} controls='false' />;
  }
});
<VideoPlayer src="video.mp4" />
```

Mixins

```
var TickTock = React.createClass({
  mixins: [SetIntervalMixin]
}
SetIntervalMixin = {
  componentWillMount: function() { .. }
}
```

Source

Thanks to http://ricostacruz.com/cheatsheets/react.html