

## THE ESSENTIALS

### 1. `React.createElement(type, props, children)`

Create a `ReactElement` with the given component class, `props` and `children`.

```
var link = React.createElement('a', {href: '#'}, "Save")
var nav = React.createElement(MyNav, {flat: true}, link)
```

### 2. `React.cloneElement(element, props, children)`

Create a new `ReactElement`, merging in new `props` and `children`.

### 3. `ReactDOM.render(element, domNode)`

Take a `ReactElement`, and render it to a DOM node. E.g.

```
ReactDOM.render(
  React.createElement('div'),
  document.getElementById('container')
)
```

### 4. `ReactDOM.findDOMNode(element)`

Return the DOM node corresponding to the given element (after `render`).

## SPECIAL PROPS

**children** is automatically added to `this.props` by `React.createElement`.

**className** corresponds to the HTML `class` attribute.

**htmlFor** corresponds to the HTML `for` attribute.

**key** uniquely identifies a `ReactElement`. Used with elements in arrays.

**ref** accepts a callback function which will be called:

1. with the component instance or DOM node on mount.
2. with `null` on unmount and when the passed in function changes.

**style** accepts an *object* of styles, instead of a string.

## PROPTYPES

Available under `React.PropTypes`. Optionally append `.isRequired`.

any      array      bool      element      func

node      number      object      string

`instanceOf(constructor)`

`oneOf(['News', 'Photos'])`

`oneOfType([propTypes, propTypes])`

## CLASS COMPONENTS

```
var MyComponent = React.createClass({
  displayName: 'MyComponent',

  /* ... options and lifecycle methods ... */

  render: function() {
    return React.createElement( /* ... */ )
  },
})
```

### Options

<b>propTypes</b>	object mapping prop names to types
<b>getDefaultProps</b>	function() returning object
<b>getInitialState</b>	function() returning object

### Lifecycle Methods

<b>componentWillMount</b>	function()
<b>componentDidMount</b>	function()
<b>componentWillReceiveProps</b>	function(nextProps)
<b>shouldComponentUpdate</b>	function(nextProps, nextState) -> bool
<b>componentWillUpdate</b>	function(nextProps, nextState)
<b>componentDidUpdate</b>	function(prevProps, prevState)
<b>componentWillUnmount</b>	function()

## COMPONENT INSTANCES

- Accessible as `this` within class components
- Stateless functional components do not have component instances.
- Serve as the object passed to `ref` callbacks
- One component instance may persist over multiple equivalent `ReactElements`.

### Properties

**props** contains any props passed to `React.createElement`

**state** contains state set by `setState` and `getInitialState`

### Methods

1. `setState(changes)` applies the given changes to `this.state` and re-renders
2. `forceUpdate()` immediately re-renders the component to the DOM