</> Component Type #1 - Class-based

```
class MyComponent extends Component {
 static propTypes   = {}
 static defaultProps = {}
  constructor(props) {
    super(props);
    this.state = { // state keys go here }
  }
  render() { }
}
```

Define a component this way when you need to maintain internal state.

</> Component Type #2 - Functional

```
function MyComponent(props) {
  return ( <div></div>);
}
```

Define a component this way when you just need to render UI (no internal state required).
Synonymous with "Presentational Component".

⟳ Lifecycle Hooks

constructor()	Called before component is mounted
render()	Never call setState() here
componentWillMount()	Before rendering (no DOM yet)
componentDidMount()	After rendering

⟳ Lifecycle Hooks (cont)

componentWillUpdate()	Can't use setState() here
componentDidUpdate()	Operate on the DOM here
componentWillReceiveProps()	Use setState() here
shouldComponentUpdate()	Skips render() if returns false

Available PropTypes

PropTypes.array	Check if prop is an array
PropTypes.string	Check if prop is a string
PropTypes.bool	Check if prop is a boolean (true/false)
PropTypes.number	Check if prop is a number
PropTypes.object	Check if prop is an object
PropTypes.node	Check if prop is anything that can be rendered in a React component
PropTypes.func	Check if prop is a function
PropTypes.element	Check if prop is a React element
PropTypes.oneOf(['M', 'F''])	Check if prop matches one of these values

Available PropTypes (cont)

`PropTypes.oneOfType([`
`  PropTypes.string,`
`  PropTypes.bool`
`])`
	Check if prop matches one of these types

`PropTypes.instanceOf(Class)`	Check if prop is an instance of a class
`PropTypes.arrayOf(Array)`	Check if prop is an array of some specific type
`PropTypes.objectOf(PropTypes.number)`	Check if prop is an object with property values of a specific type

`PropTypes.shape({`
`  potato: PropTypes.object,`
`  turkey: PropTypes.string`
`})`
	Check if prop object has a particular "shape"

`PropTypes.any`	Check if anything at all is passed

☑ Reusability Checklist (ES6)

Step #1 - Define your component and import PropTypes from the 'prop-types' package

```
import React, { Component } from 'react';
import PropTypes         from 'prop-types';

class MyComponent extends Component {
  static propTypes   = {}
  static defaultProps = {}

  constructor(props) {
    super(props);
  }

  render() {}
}
```

Step #2 - Define the props & what type they are

```
...
static propTypes: {
  className  : PropTypes.string
  isClosed   : PropTypes.bool
  email      : PropTypes.string
}
...
```

Step #3 - Determine which props are required or optional

```
...
static propTypes: {
  className  : PropTypes.string
  isClosed   : PropTypes.bool
  email      : PropTypes.string.isRequired
}
...
```

Step #4 - Define the defaults (if applicable)

```
...
static defaultProps: {
  className : "btn"
  isClosed  : true
}
...
```

By **hackingbeauty**
(hackingbeauty)

cheatography.com/hackingbeauty/
hackingbeauty.com

Published 7th May, 2017.
Last updated 7th May, 2017.
Page 2 of 3.

Cheatography

React Reusability Cheat Sheet
by hackingbeauty (hackingbeauty) via cheatography.com/37001/cs/11658/

</> You can custom validate a prop

```
...
static propTypes: {
  email: ((props, propName) => {
    const regex = /^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/;
    if (!regex.test(props[propName])) {
     return new Error('Invalid email!');
    }
  })
}
...
```

Other APIs

`setState(updater, [callback])`	Component will re-render with new state
`forceUpdate()`	Calls `render()`, skips `shouldComponentUpdate()`

Next Steps

Knowing how to create reusable components is step 1.

Then you need to learn how to architect professional apps using React & Redux.

Learn how at www.singlepageapplication.com