REACT JS

INTRODUCTION

CONTENT

WHAT IS REACT?

- A JavaScript Library For Building User Interfaces
- Renders your UI and responds to events.
- It also uses the concept called Virtual DOM, creates an in-memory data structure cache, enumerates the resulting differences, and then updates the browser's displayed DOM efficiently.
- One of the unique features of React.js is not only it can perform on the client side, but it can also be rendered on the server side, and they can work together interoperably.

WHAT IS REACT?

Angular has

- modules
- controllers
- directives
- scopes
- templating
- linking functions
- filters
- dependency injection

WHAT IS REACT?

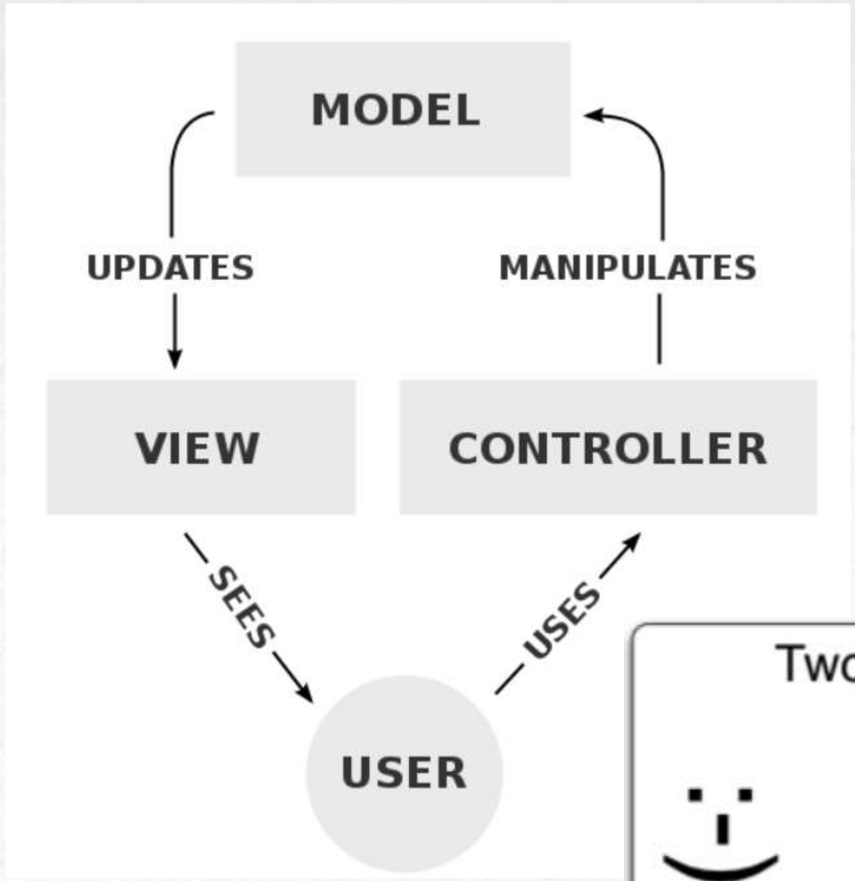
~~Angular~~ React has **JUST COMPONENT**

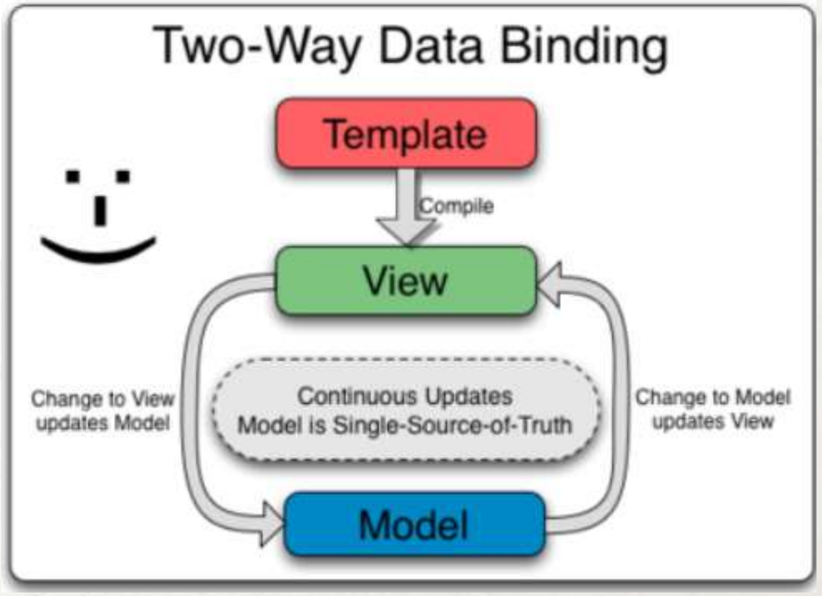
- ~~modules~~
- ~~controllers~~
- ~~directives~~
- ~~scopes~~
- ~~templating~~
- ~~linking functions~~
- ~~filters~~
- ~~dependency injection~~

WHAT IS REACT?

#2 Single Source of Truth

MVC proposes that your **Model is the single source of truth**— all state lives there. Views are **derived** from the Model, and must be **kept in sync**. When the Model changes, so does the View.

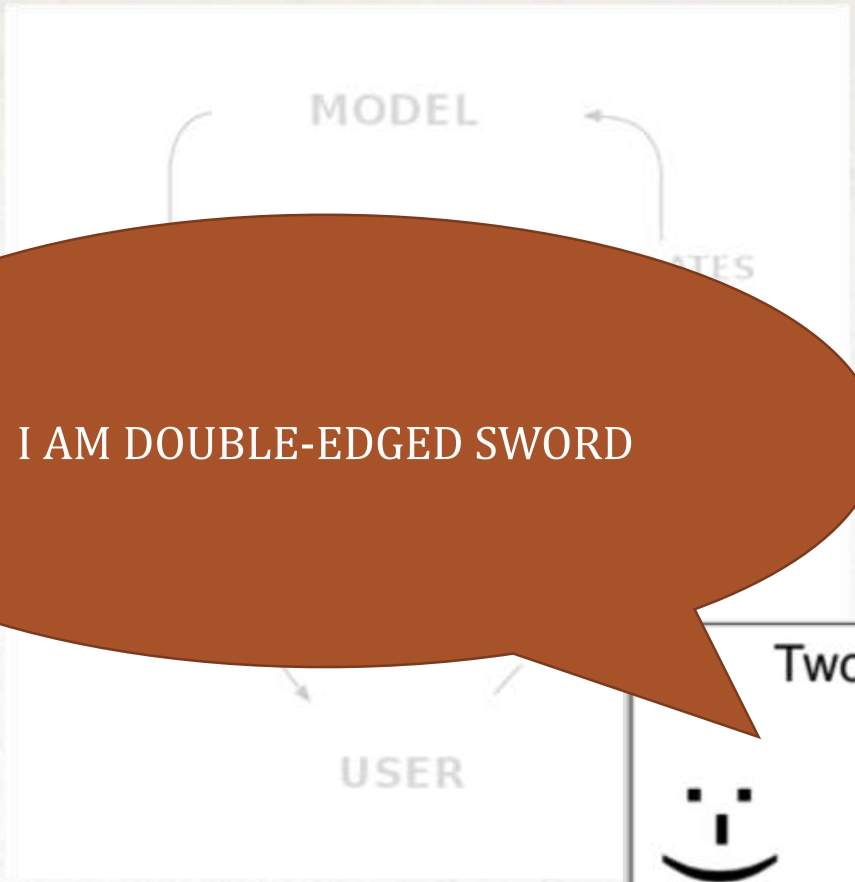



Two-Way Data Binding
Template
Compile
View
Continuous Updates
Model is Single-Source-of-Truth
Change to View updates Model
Change to Model updates View
Model

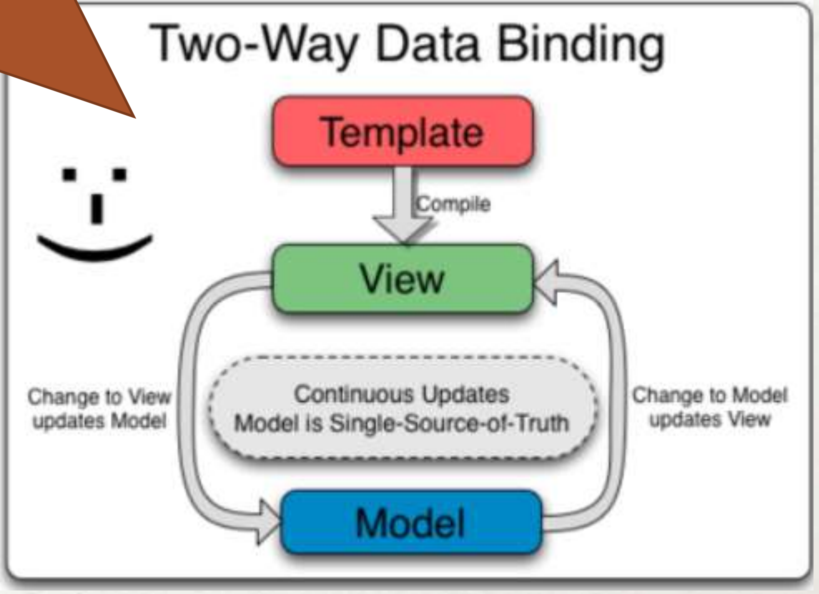
WHAT IS REACT?

#2 Single Source of Trut

MVC proposes that your M **is the single source of trut** all state lives there. Views are **derived** from the Model, and must be **kept in sync**. When the Model changes, so does the View.



I AM DOUBLE-EDGED SWORD


Two-Way Data Binding
Template
Compile
View
Continuous Updates
Model is Single-Source-of-Truth
Model
Change to View updates Model
Change to Model updates View

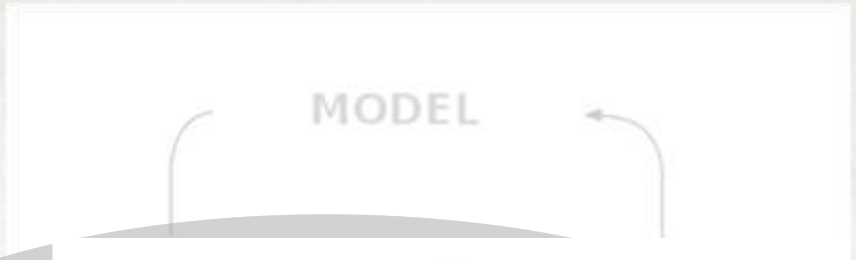
WHAT IS REACT?

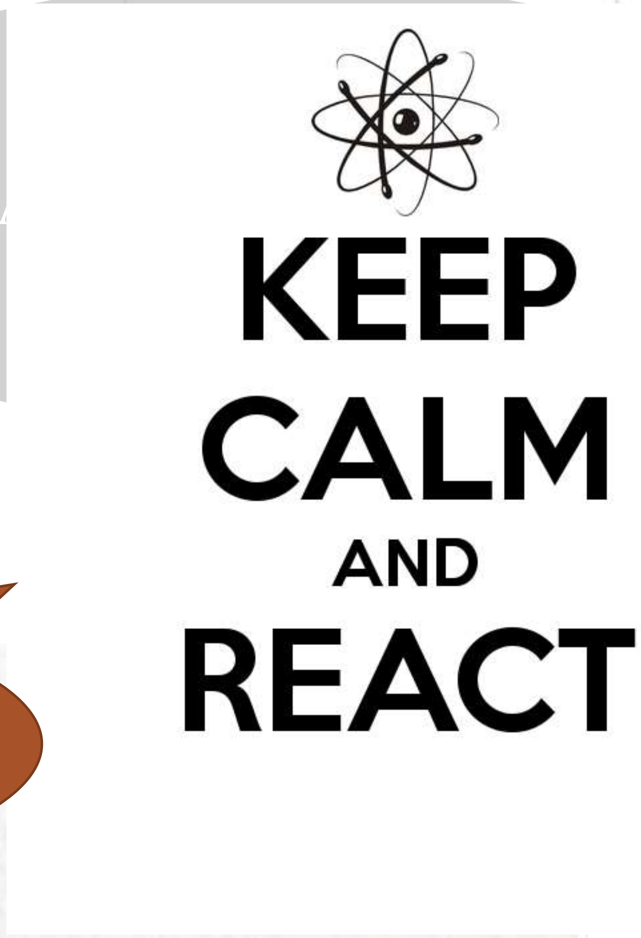
#2 Single Source of Truth

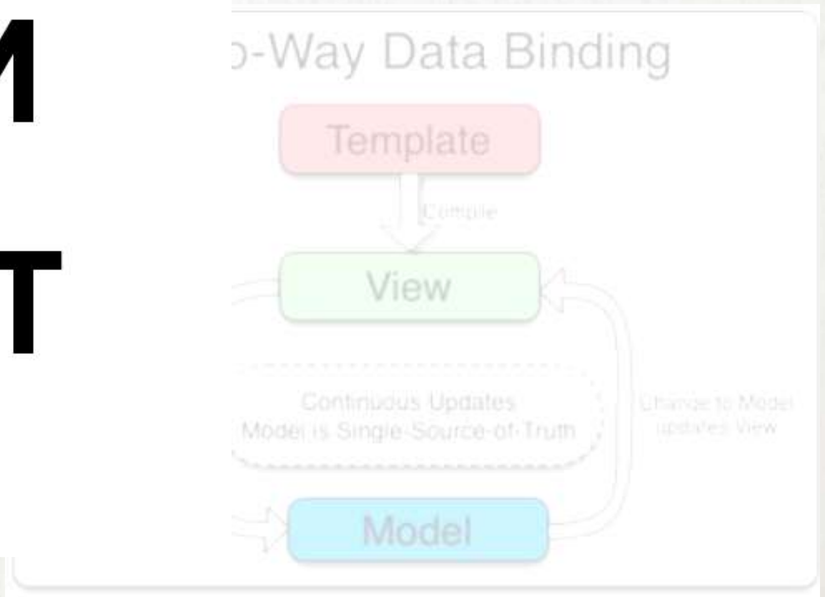
MVC proposes that your **Model is the single source of truth** — all state lives there. Views are **derived** from the Model, and must be **kept in sync**. When the Model changes, so does the View.

I A

Only render when state changed

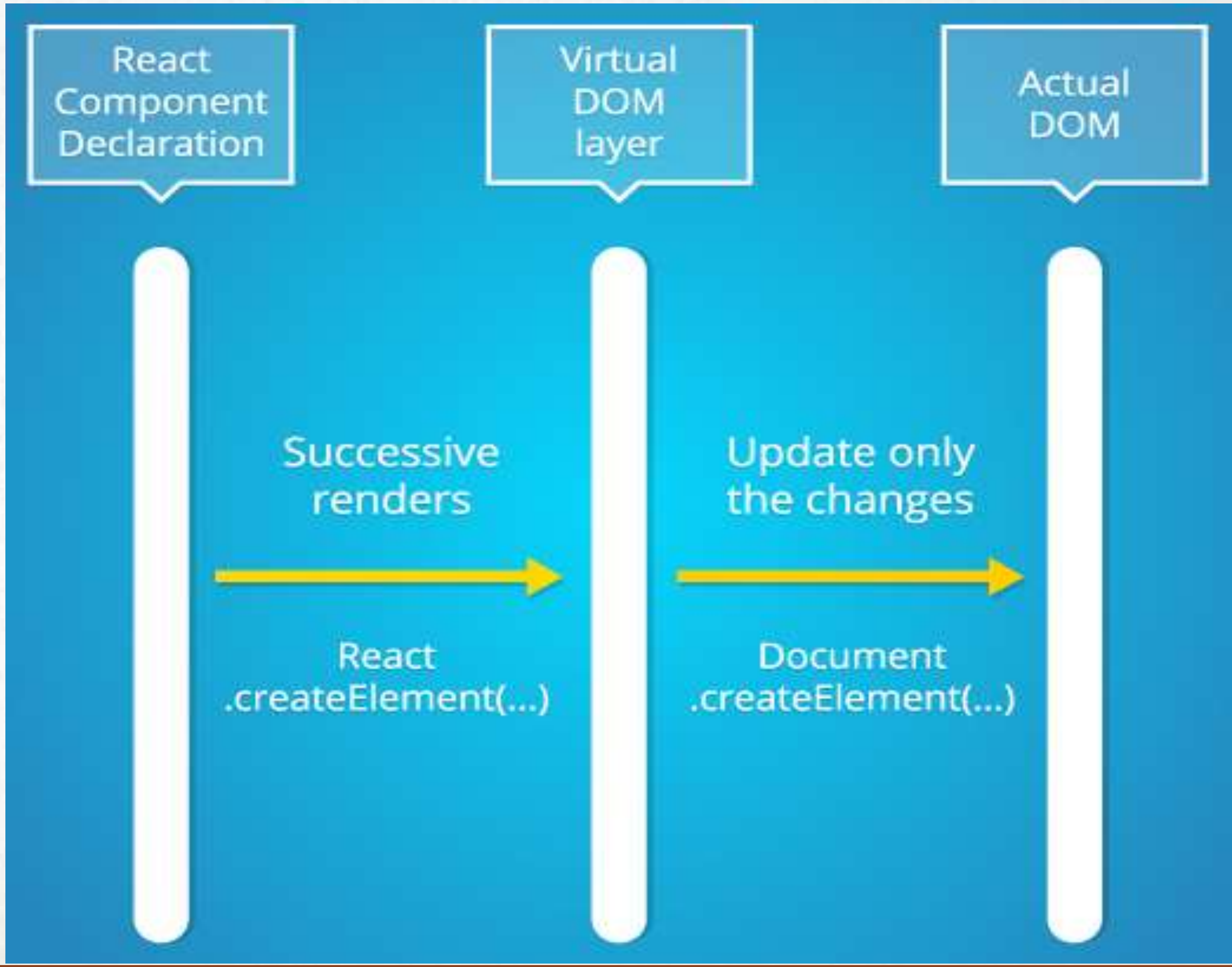






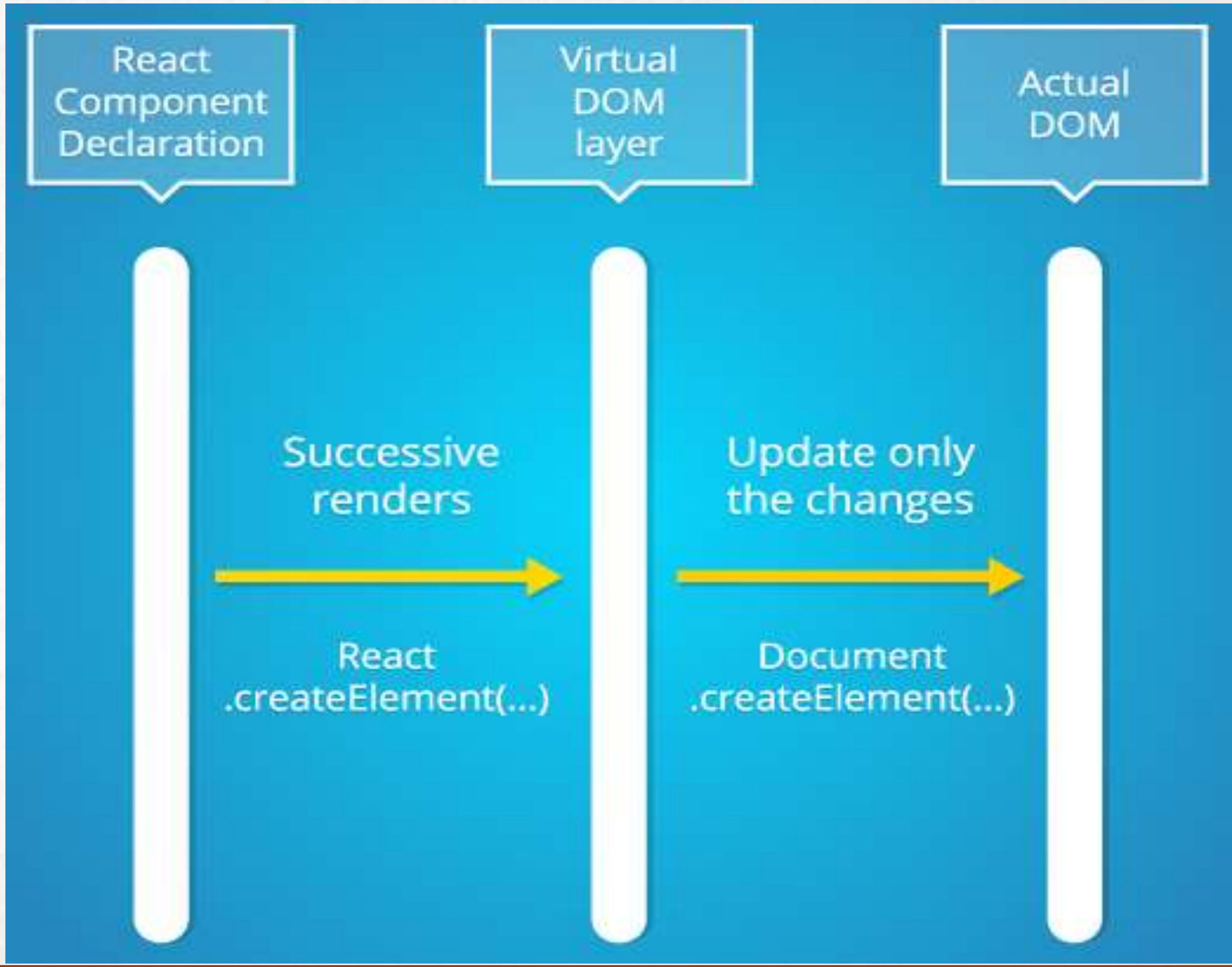
WHAT IS REACT? – VIRTUAL DOM

- Manipulate DOM is high cost.
- React first assembles the *entire* structure of your app **in-memory**, using those objects. Then, it **converts** that structure into **actual DOM nodes** and inserts them in your browser’s DOM.


React Component Declaration
Virtual DOM layer
Actual DOM
Successive renders
React .createElement(...)
Update only the changes
Document .createElement(...)

WHAT IS REACT? – VIRTUAL DOM

```
<script>
 var helloEl = React.createElement(
  'div',
  { className: 'hello' },
  'Hello, world!`
 );
 React.render(helloEl,document.body);
</script>
```


React Component Declaration
Virtual DOM layer
Actual DOM
Successive renders
React .createElement(...)
Update only the changes
Document .createElement(...)

JSX

- JSX = Javascript + XML.

```
const element = <h1>Hello, world!</h1>;
```

JSX

```
<script>
 var helloEl = React.createElement('div', { className: 'hello' }, 'Hello,
                world!');
 React.render(
  helloEl,
  document.body
 );
</script>
```

```
<script type="text/jsx">
 var helloEl = <div className: "hello">Hello, world!</div>;
 React.render(
  helloEl,
  document.body
 );
</script>
```

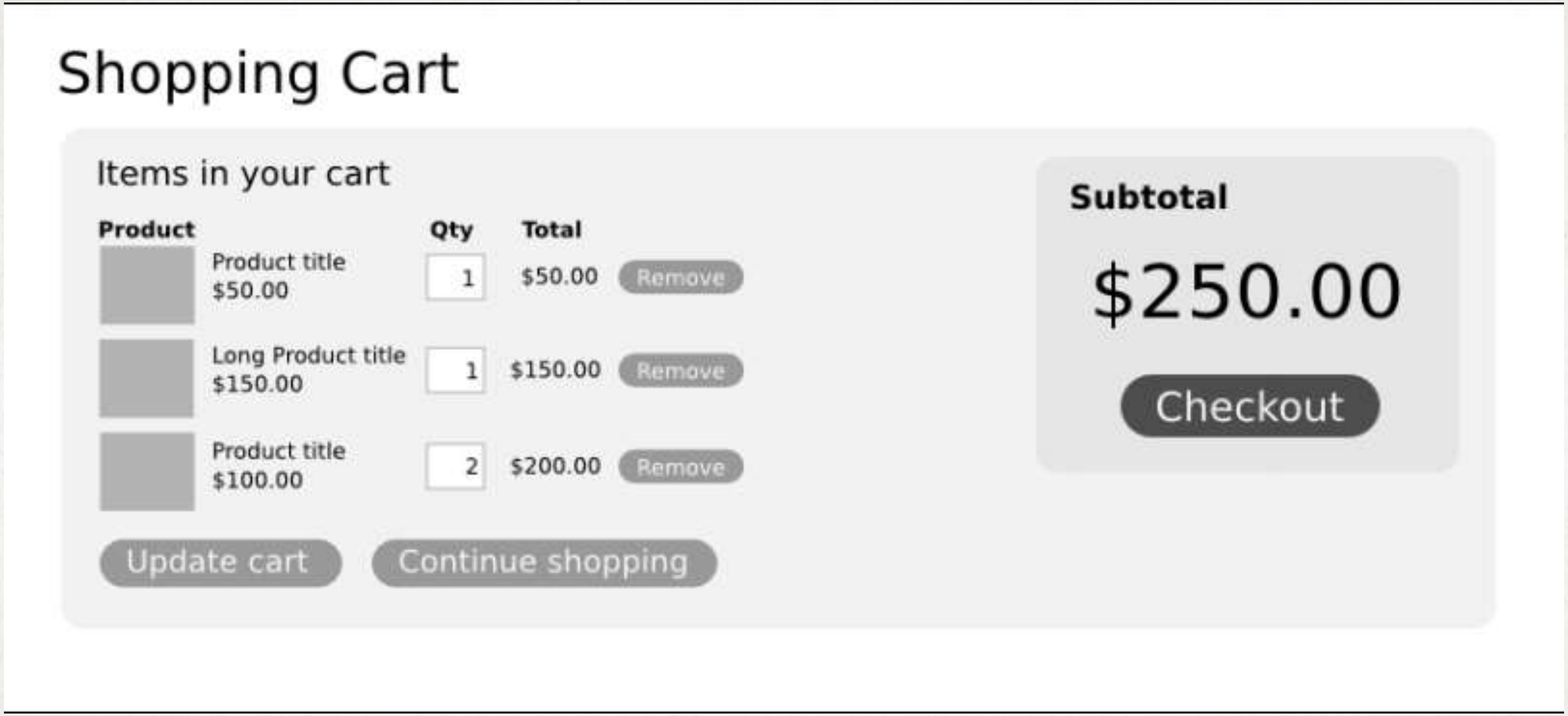
JSX

```
<script>
 var helloEl = React.createElement('div', { className: 'hello' }, 'Hello,
                world!');
 React.render(
  helloEl,
  document.body
 );
</script>
```

```
<script type="text/jsx">
 var helloEl = <div className: "hello">Hello, world!</div>;
 React.render(
  helloEl,
  document.body
 );
</script>
```

COMPONENT

- Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.
- Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.



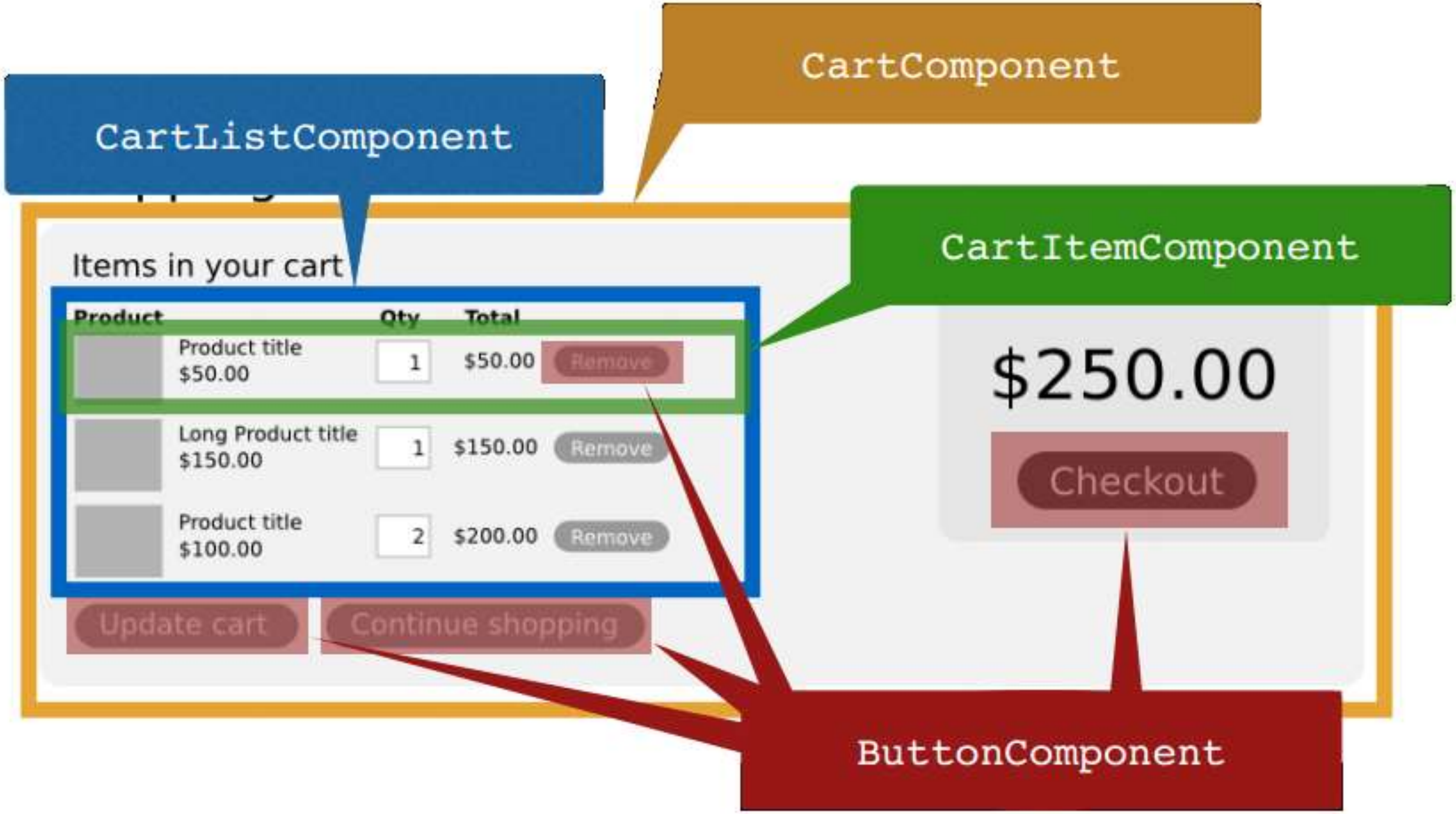
Shopping Cart

Items in your cart

Product		Qty	Total	
	Product title $50.00	1	$50.00	Remove
	Long Product title $150.00	1	$150.00	Remove
	Product title $100.00	2	$200.00	Remove

Subtotal

$250.00


CartComponent
CartListComponent
CartItemComponent
ButtonComponent
Items in your cart
Product
Qty
Total
Product title
$50.00
1
$50.00
Remove
Long Product title
$150.00
1
$150.00
Remove
Product title
$100.00
2
$200.00
Remove
Update cart
Continue shopping
$250.00
Checkout

COMPONENT

```
class TodoInput extends React.Component {

  render() {

    return (
      <div className="form-inline">
        <input className="form-control" type="text" value={this.state.content}
               onChange={this.updateState}/>
      </div>
    );
  }
}
```

COMPONENT - PROPS

- Props is what you pass into the Component via attributes.
- Props is the only way to input data. (Or you can use Redux).
- Props are immutable.
- Container component will define data that can be changed
- Child Component will received data from parent component via props.

COMPONENT - PROPS

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>{this.props.headerProp}</h1>
        <h2>{this.props.contentProp}</h2>
      </div>
    );
  }
}
export default App;
```

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.jsx';

ReactDOM.render(
  <App headerProp="Header from props..."
contentProp="Content from props..."/>,
document.getElementById('app')
);

export default App;
```

COMPONENT - STATE

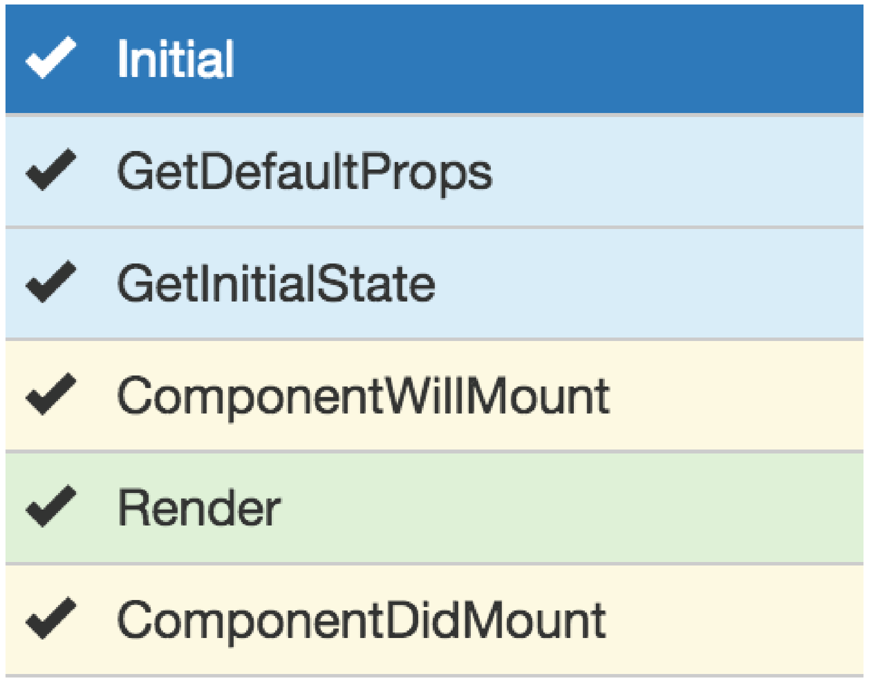
- Private data of component
- When change -> Re-render Component
- Can’t read from outside Component

```
class TodoInput extends React.Component {
  constructor(props) {
    super(props); //Call this function because 'this' is not allowed before super().
    this.state = {
      content: ''
    };
    this.addTodo = this.addTodo.bind(this);
  }

  updateState(e) {
    this.setState({content: e.target.value});
  }
  addTodo() {
    // We of course not declare onSave function of this component at parent component
    // Refer to: Body.jsx for more information
    // We declare this onSave at mapDispatchToProps function
    this.props.onSave.call(this, this.state.content, this.props.todo && this.props.todo.id || null);
    this.setState({
      content: ''
    })
  }
  render() {
    return (
      <div className="form-inline">
        <div className="form-group">
          <input className="form-control" type="text" value={this.state.content} onChange={this.updateState}/>
        </div>
      </div>
    );
  }
}
```

REACT COMPONENT LIFECYCLE

- **React** enables to create components by invoking the React.createClass() method which expects a *render* method and triggers a lifecycle that can be hooked into via a number of so called lifecycle methods.
- This short article should shed light into all the applicable functions.
- Understanding the component lifecycle will enable you to perform certain actions when a component is created or destroyed. Further more it gives you the opportunity to decide if a component should be updated in the first place and to react to props or state changes accordingly.

THE LIFECYCLE - INITIALIZATION



✓ Initial

✓ GetDefaultProps

✓ GetInitialState

✓ ComponentWillMount

✓ Render

✓ ComponentDidMount

- Occurs when the component is created.

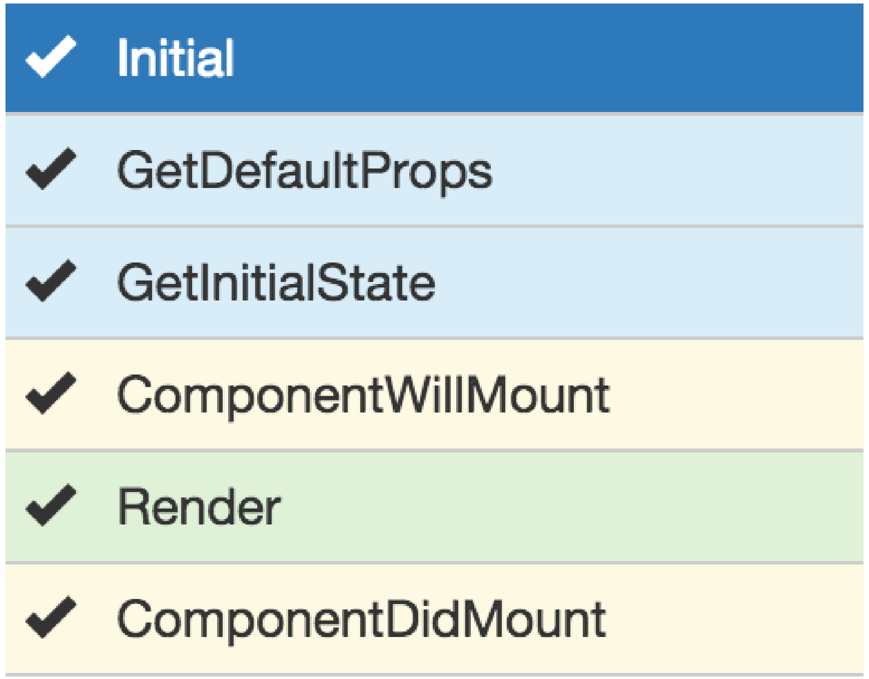
```
var Greeting = React.createClass({
  propTypes: {
    name: React.PropTypes.string
  },

  getDefaultProps: function () {
    return {
      name: 'Mary'
    };
  },

  getInitialState: function () {
    return {
      helloSentence: 'Hello'
    }
  }

  // ...
});
```

THE LIFECYCLE - INITIALIZATION

- getDefaultProps and getInitialState not exists when define Component as Class ES6.

```
Greeting.defaultProps = {
  name: 'Mary'
};
```

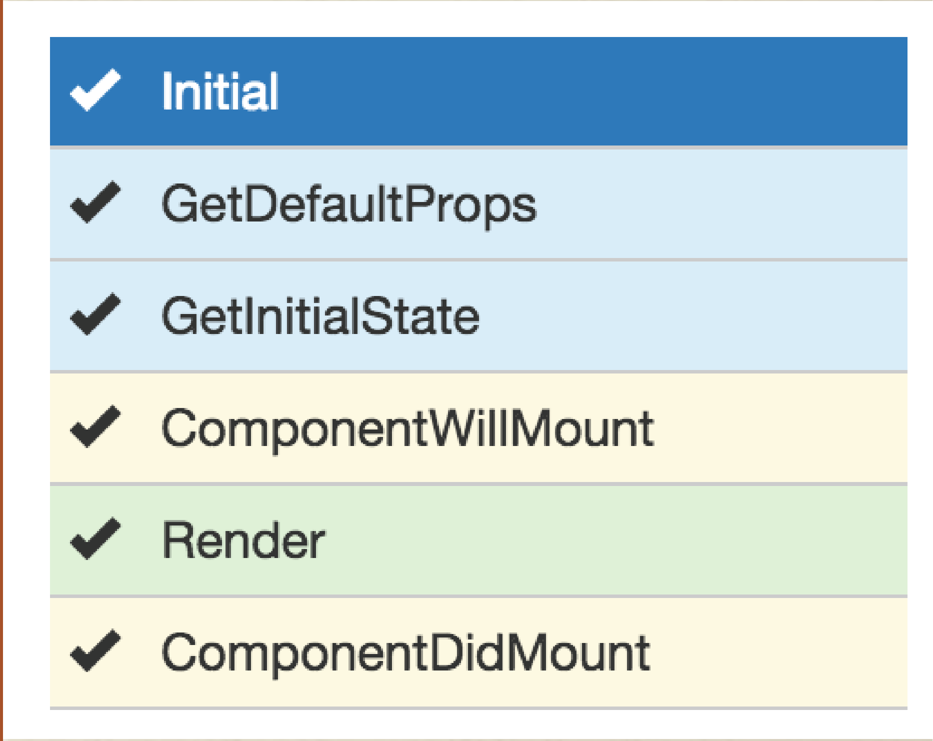
```
constructor(props){
  super(props);
  this.state = {
    name: 'Mary'
  }
}
```



✓ Initial
✓ GetDefaultProps
✓ GetInitialState
✓ ComponentWillMount
✓ Render
✓ ComponentDidMount

THE LIFECYCLE - INITIALIZATION

- Inside ComponentWillMount, setting state won’t trigger re-render whole component.
- We CAN NOT modify state in render method.
- DOM Manipulation is only permitted inside componentDidMount method.



✓ Initial

✓ GetDefaultProps

✓ GetInitialState

✓ ComponentWillMount

✓ Render

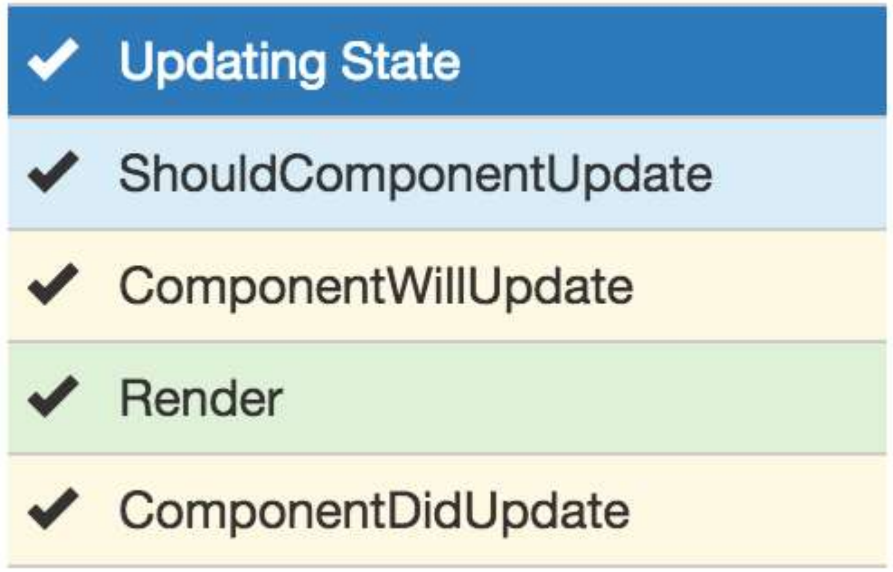
✓ ComponentDidMount

THE LIFECYCLE - STATE CHANGES

- Occur when state is changed (via this.setState(..)) except inside componentWillMount methods

```
shouldComponentUpdate: function(nextProps, nextState){
  // return a boolean value
  return true;
}
```

- shouldComponentUpdate returning false results in followed methods won’t be triggerd also.
- shouldComponentUpdate won’t triggered in the initial phase or when call forceUpdate().
- Current State of Component DID NOT have new value,



✓ Updating State
✓ ShouldComponentUpdate
✓ ComponentWillUpdate
✓ Render
✓ ComponentDidUpdate

THE LIFECYCLE - PROPS CHANGES

- Occurs when data passed from parent component to child component changed (via props).

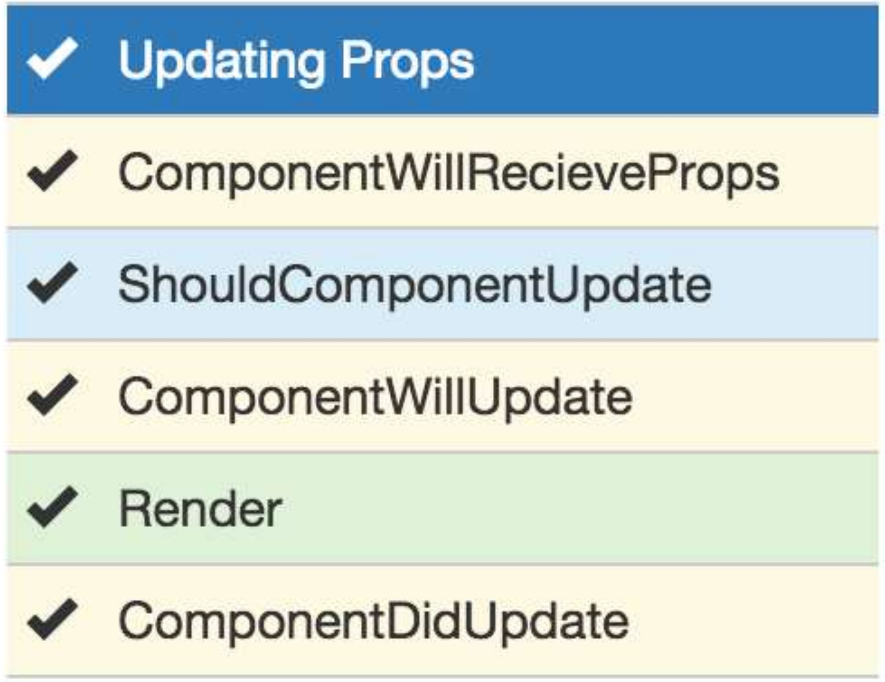
Props Change ➔ componentWillReceiveProps trigged

NOT

Props Change ⇔ componentWillReceiveProps trigged

- Changing states in *ComponentWillReceiveProps* DID NOT trigger re-render component.

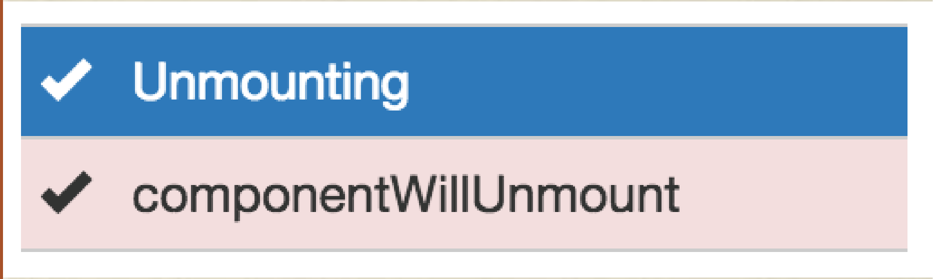
```
componentWillReceiveProps: function(nextProps)
{
  this.setState({
    // set something
  });
}
```



✓ Updating Props
✓ ComponentWillRecieveProps
✓ ShouldComponentUpdate
✓ ComponentWillUpdate
✓ Render
✓ ComponentDidUpdate

THE LIFECYCLE - UNMOUNTING

- Used to clean up data



✓ Unmounting

✓ componentWillUnmount

PROS & COS OF REACT.JS

THE GOOD POINTS:

- **React.js is extremely efficient**
  - Virtual DOM
- **It makes writing Javascript easier**
  - React.js uses a special syntax called JSX
- **It gives you out-of-the-box developer tools**
  - React.js chrome extension
- **It's awesome for SEO**

  - Server rendering
- **UI Test Cases**

THE BAD:

- React.js is only a view layer.
- There is a learning curve for beginners who are new to web development.
- Library size. (~ Angular)

Why you should use React.js:

- React.js works great for teams, strongly enforcing UI and workflow patterns.
- The user interface code is readable and maintainable.
- Componentized UI is the future of web development, and you need to start doing it now.
- And also, there is now a lot of demand for developers with ReactJS experience.

Why you should NOT use React.js:

- **Slow you down tremendously** at the start.
- You will reinvent a lot of wheels.

OUR NEXT STEPS

- Flux/Redux
- React Native

REFERENCES

- https://firstdoit.com/how-i-learned-to-stop-worrying-and-love-react-4e22b0bb6c2a#.vt2mjxu6s
- https://offroadcode.com/journal/news/reactjs-whats-it-all-about/
- http://sixrevisions.com/javascript/why-i-ditched-angular-for-react/
- https://github.com/hpphat92/my-todo-reactjs
- https://hpphat.wordpress.com/category/web-dev/react/
- http://blog.andrewray.me/reactjs-for-stupid-people/
- \\192.168.1.240\share\Phat.Hong\Reactjs

Q&A

Thank you for listening!