INTRODUCTION TO REACTJS

ROHIT GOYAL

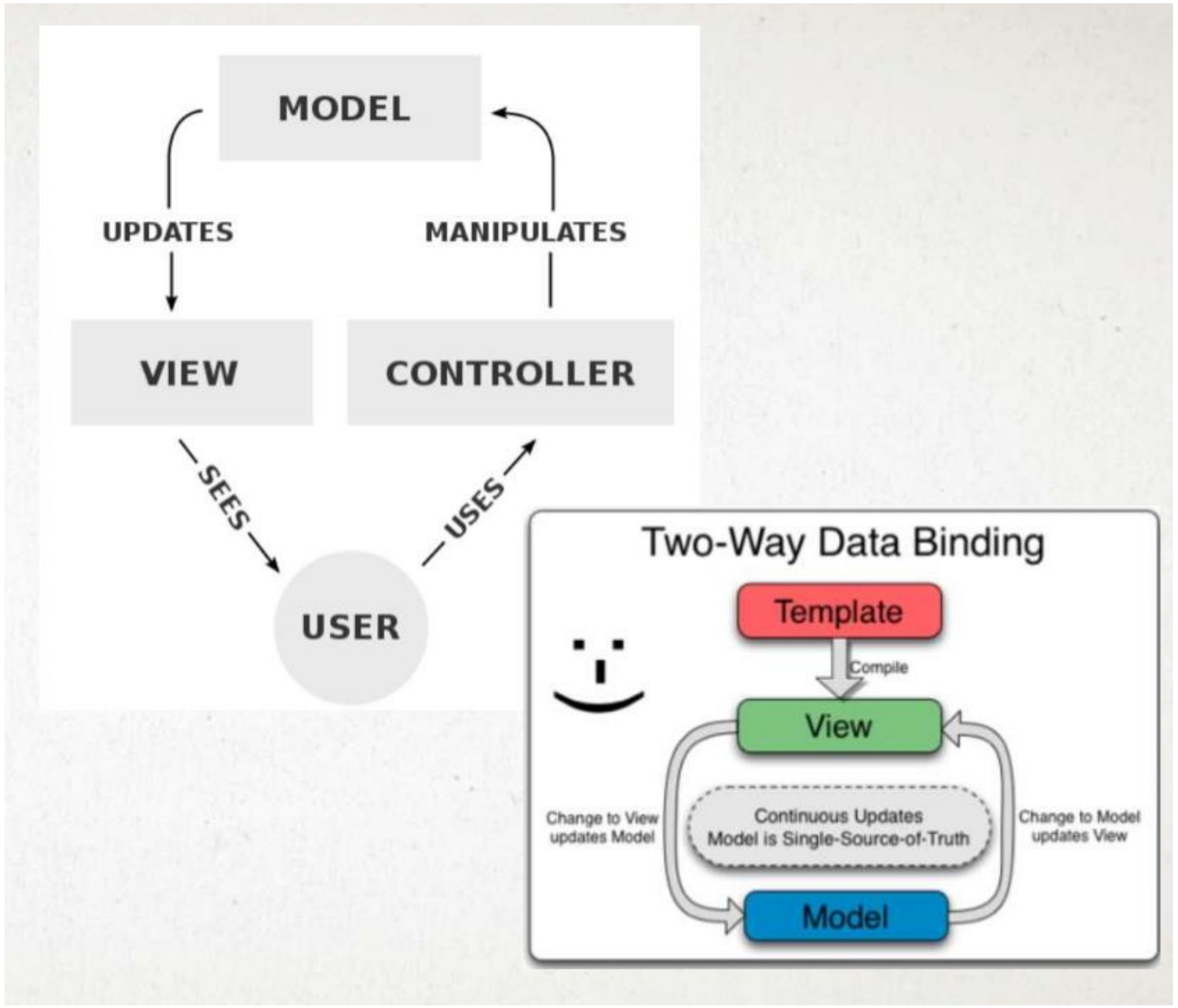
CONTENT

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

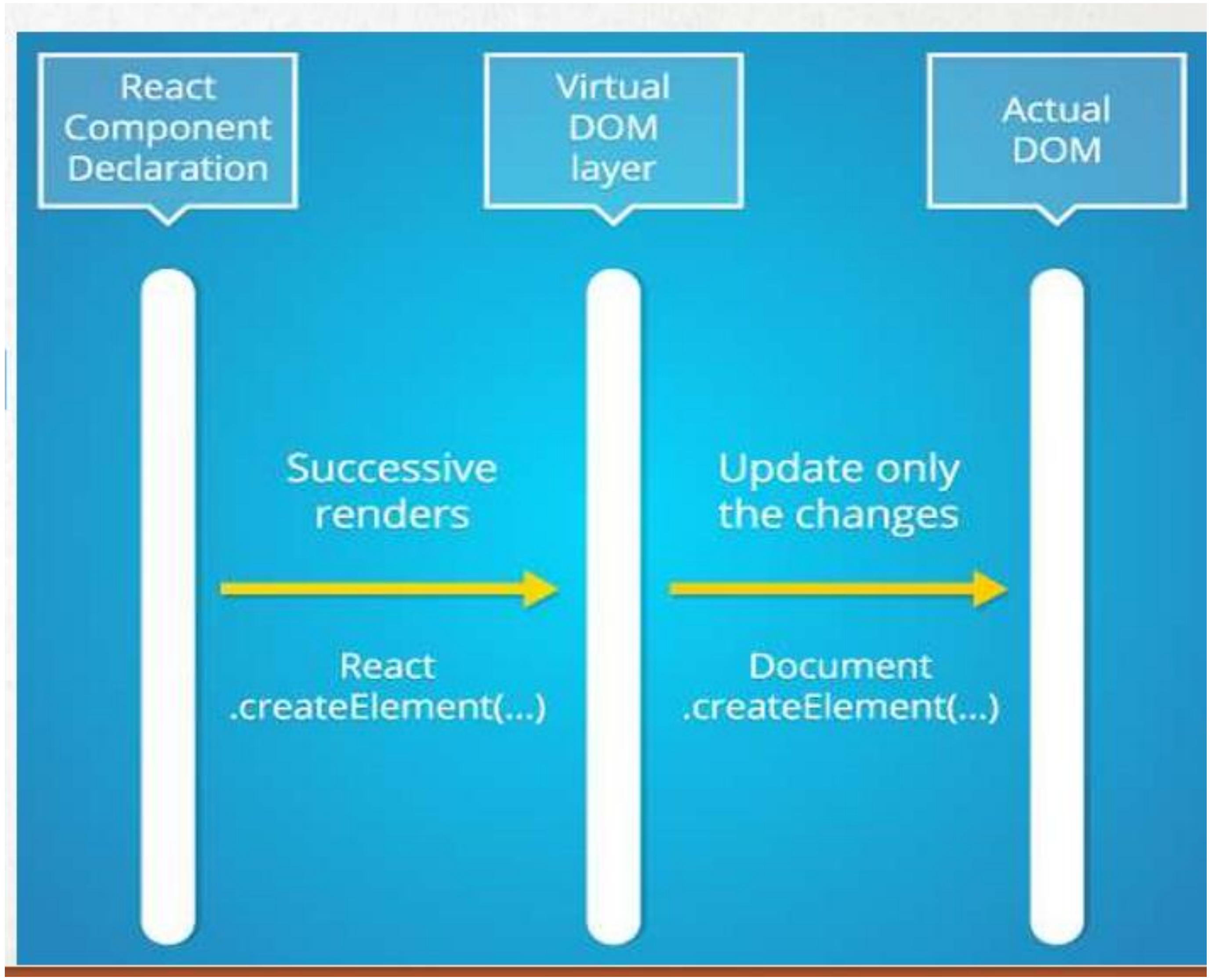
WHAT IS REACT?

- React has JUST COMPONENT
- 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.
  - Only render when state changed

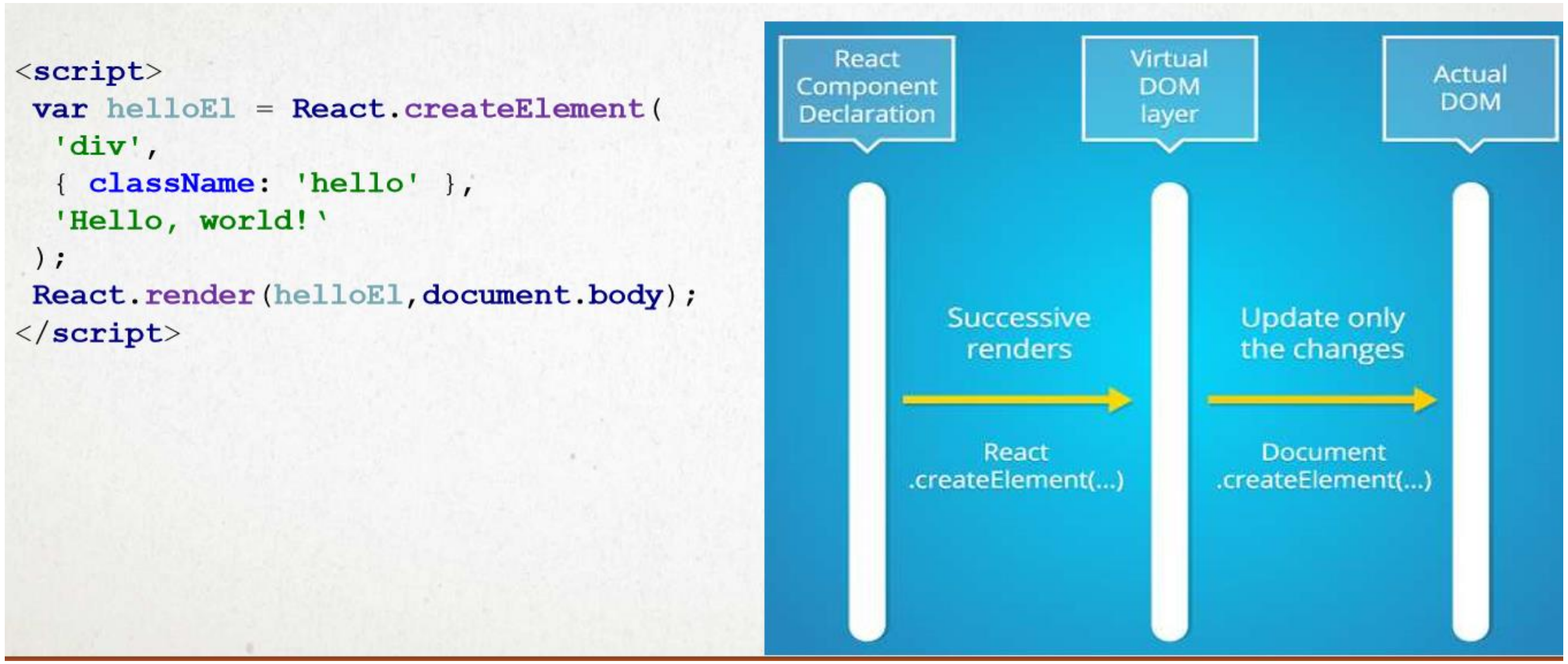

MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER
Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates Model is Single-Source-of-Truth
Change to Model updates View
Model

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



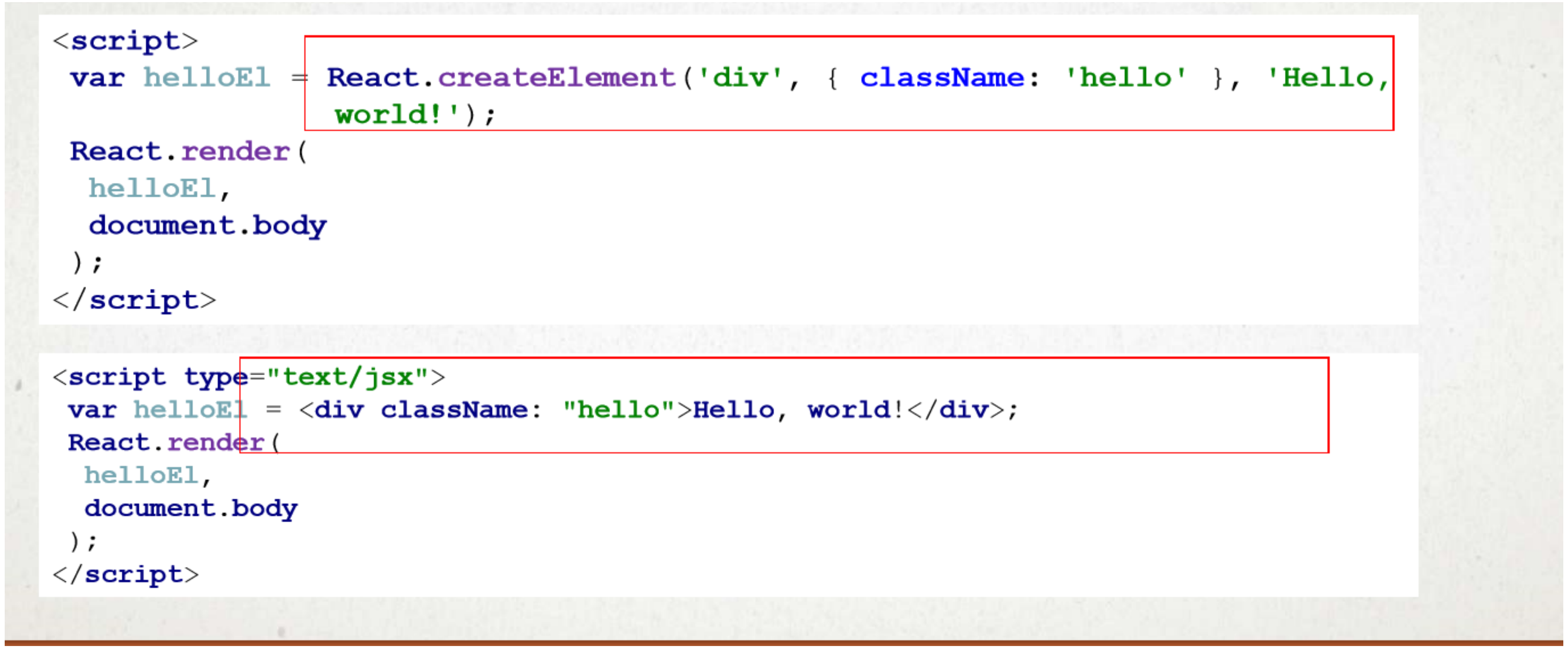
```
<script>
 var helloEl = React.createElement(
  'div',
  { className: 'hello' },
  'Hello, world!'
 );
 React.render(helloEl,document.body);
</script>
```

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



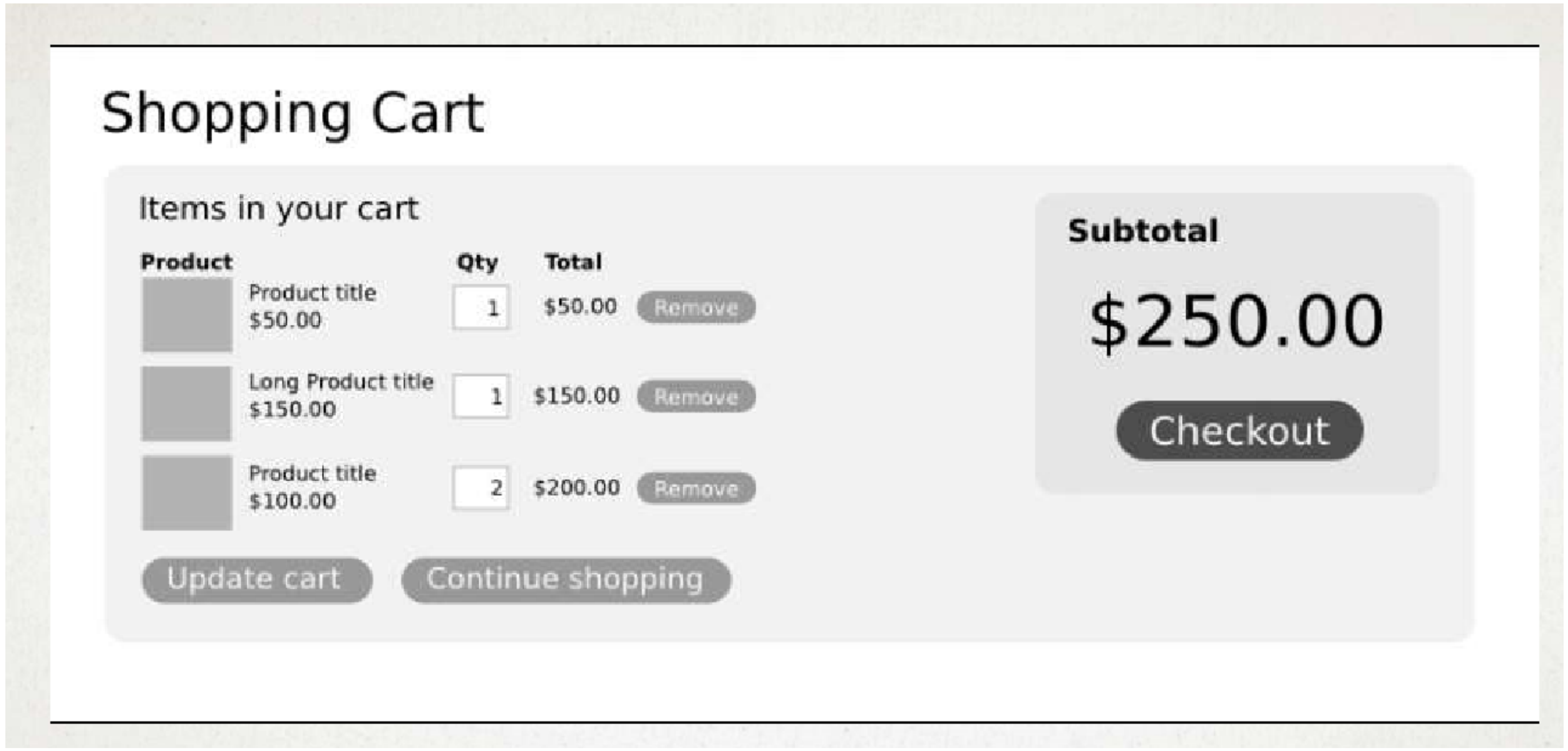
```
<script>
 var helloEl = React.createElement('div', { className: 'hello' }, 'Hello,
  world!');
 React.render(
  helloEl,
  document.body
 );
</script>
```

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

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.

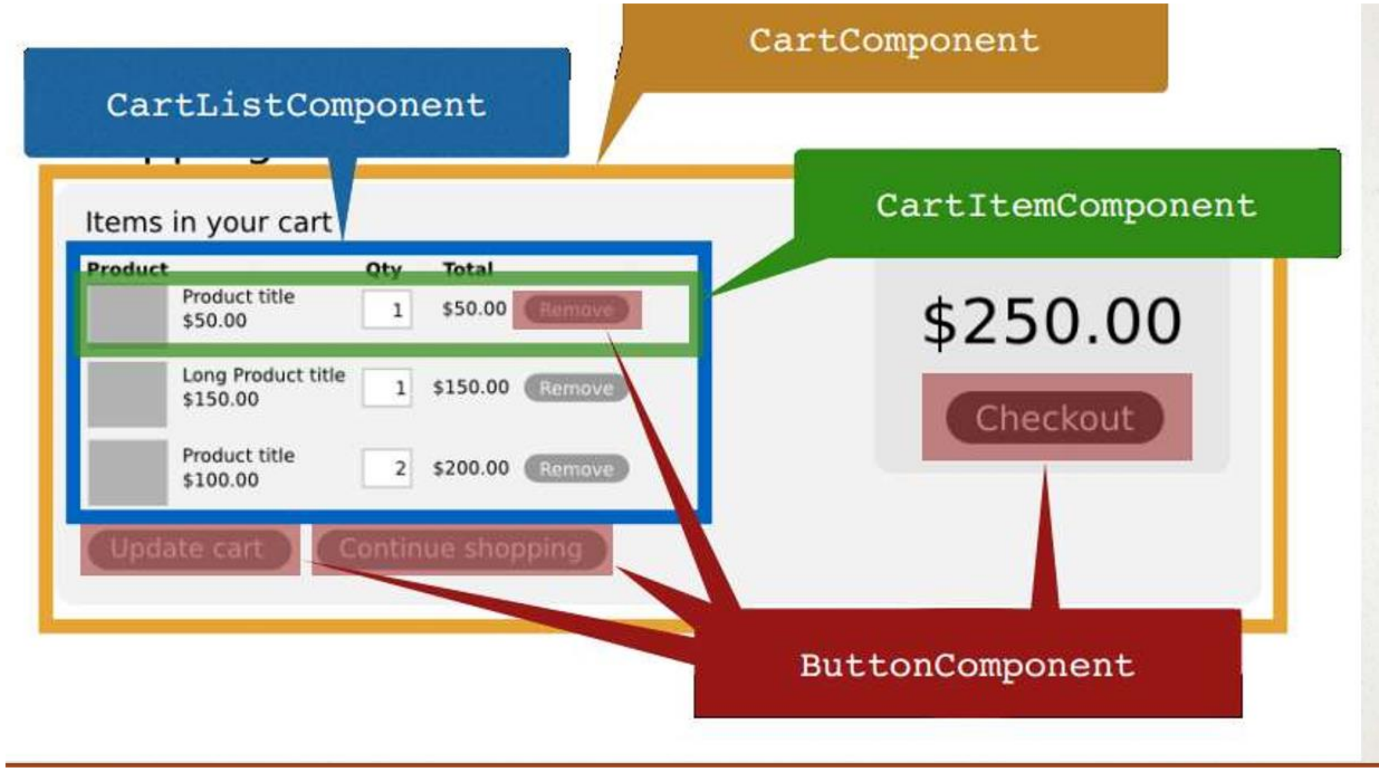
COMPONENT


Shopping Cart

Items in your cart

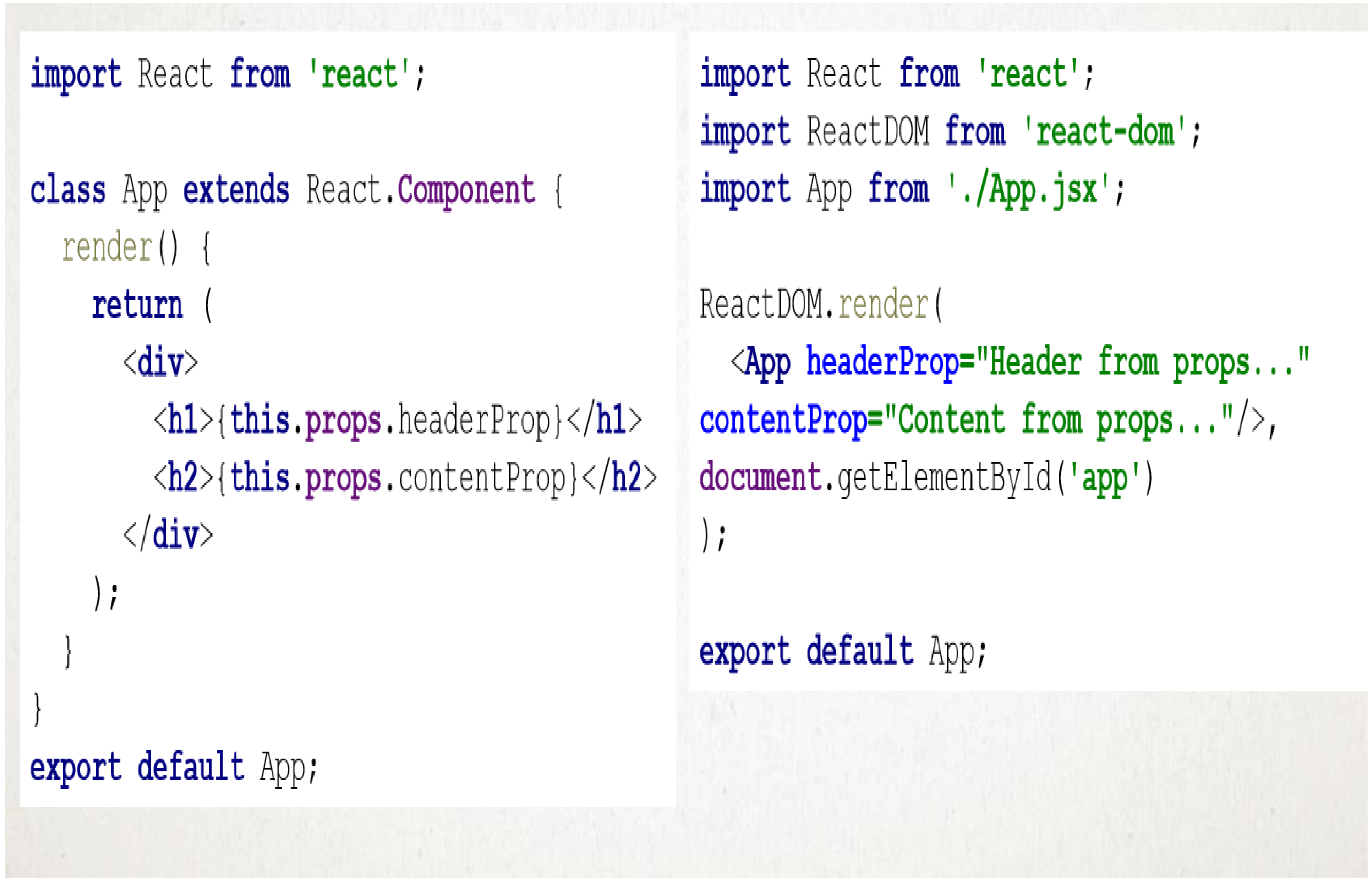
Product		Qty	Total	
	Product title $50.00	1	$50.00	Remove
	Long Product title $150.00	1	$150.00	Remove
	Product title $100.00	2	$200.00	Remove

COMPONENT


CartComponent
CartListComponent
Items in your cart
CartItemComponent
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
$250.00
Checkout
Update cart
Continue shopping
ButtonComponent

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 receive data from parent component via props.

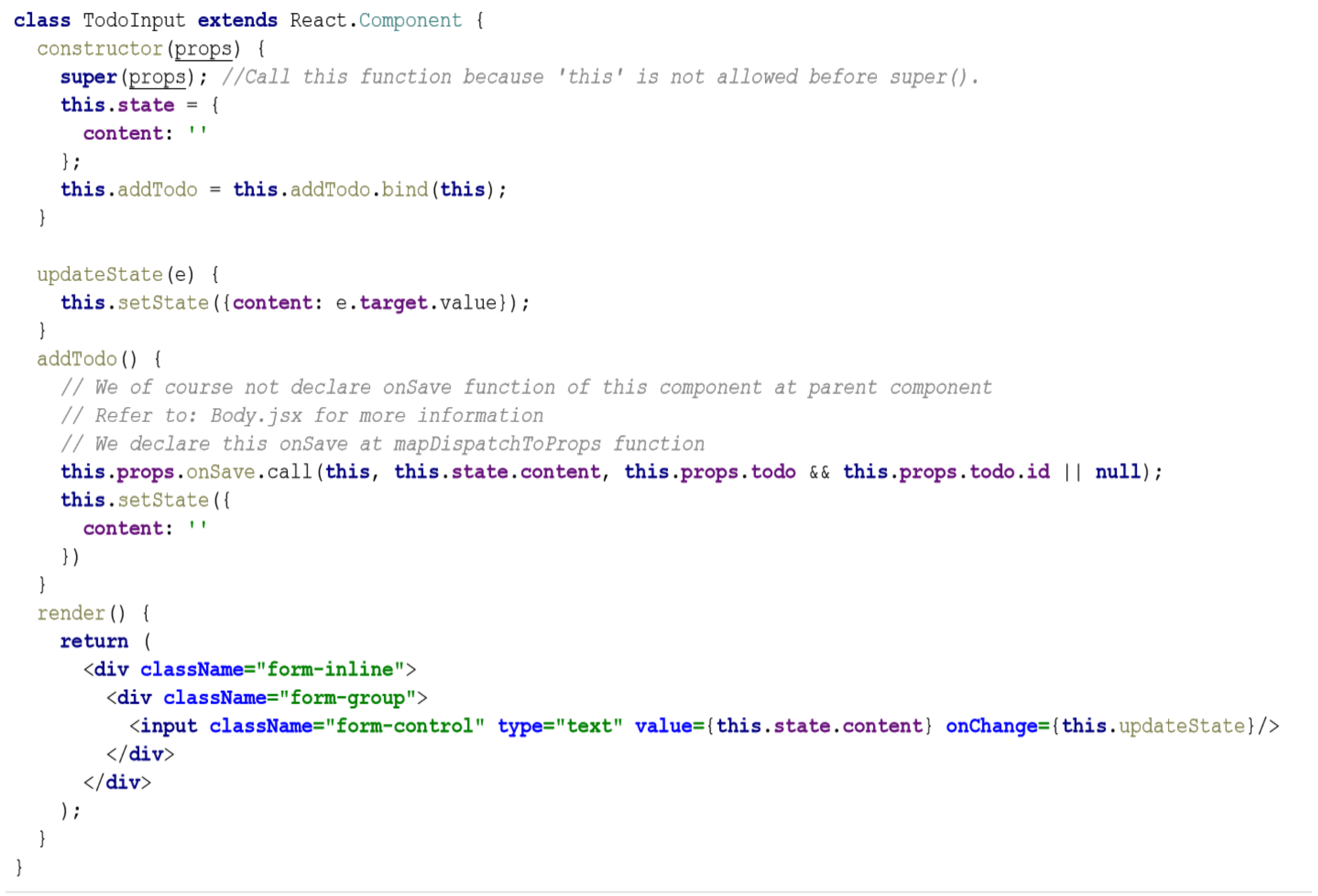


```
import React from 'react';

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



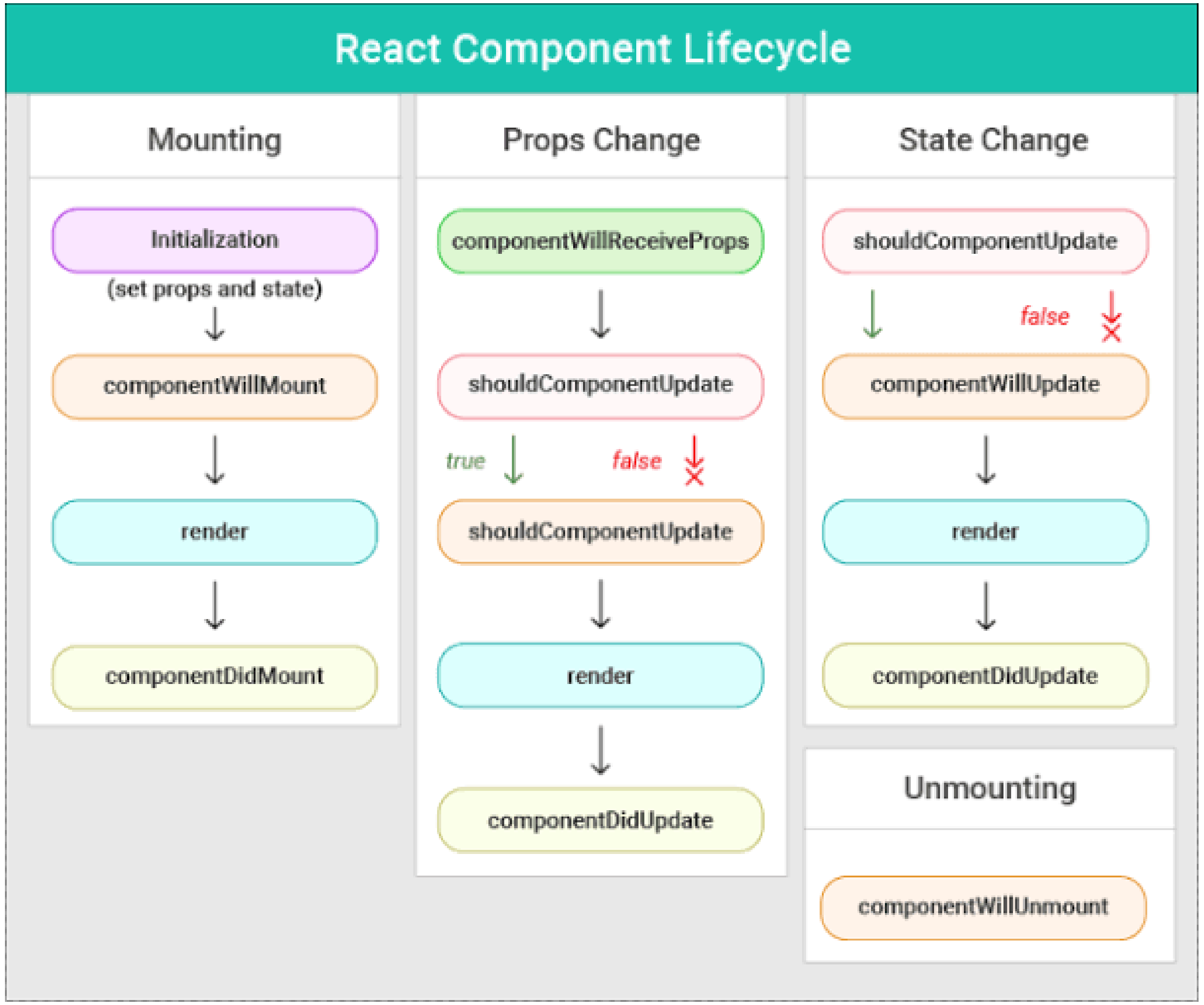
```
class TodoInput extends React.Component {
  constructor(props) {
    super(props); //Call this function because 'this' is not allowed before super().
    this.state = {
      content: ''
    };
    this.addTodo = this.addTodo.bind(this);
  }

  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.
- Understanding the component lifecycle will enable you to perform certain actions when a component is created or destroyed. Furthermore 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

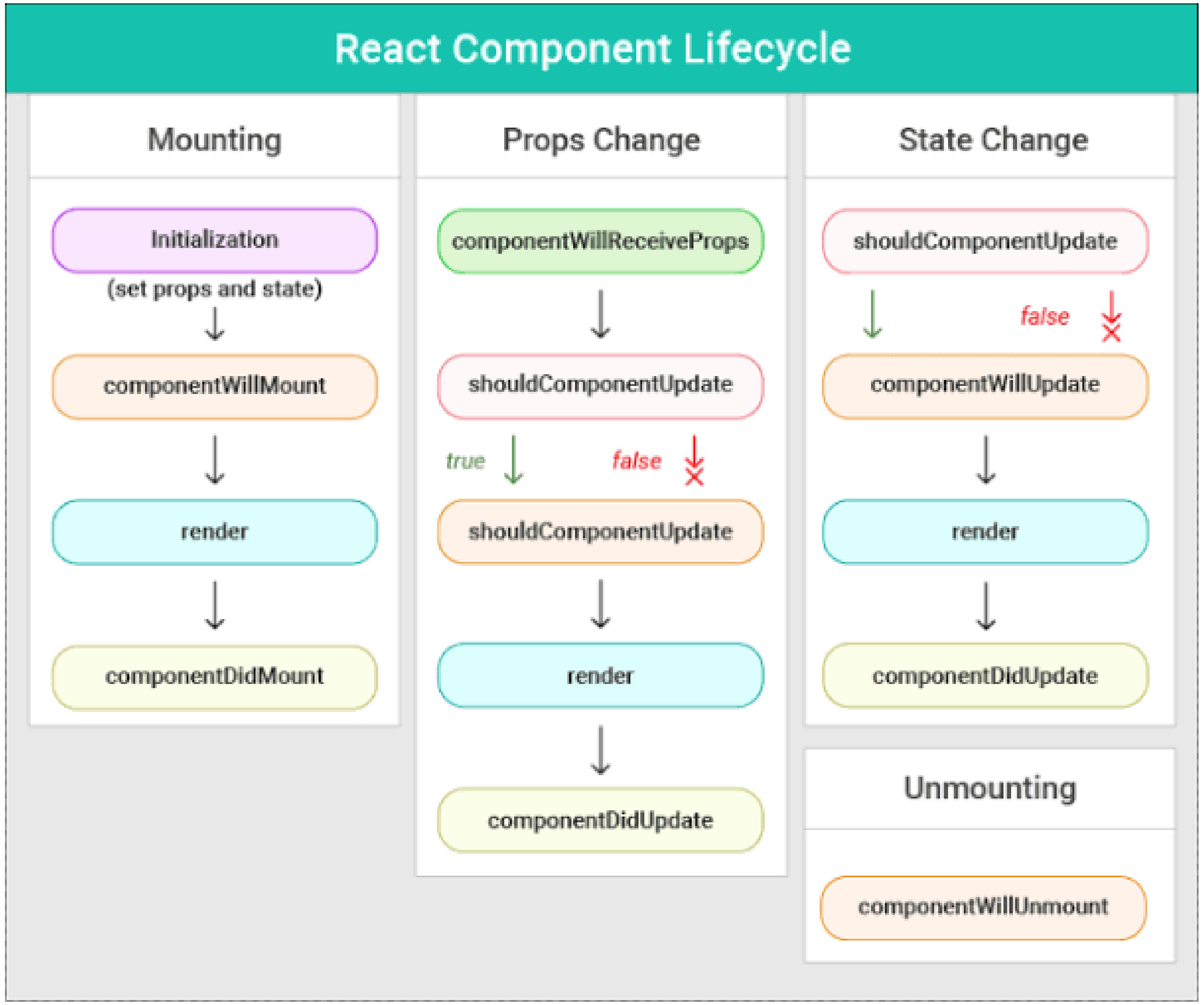
REACT COMPONENT LIFECYCLE - MOUNTING

- 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.



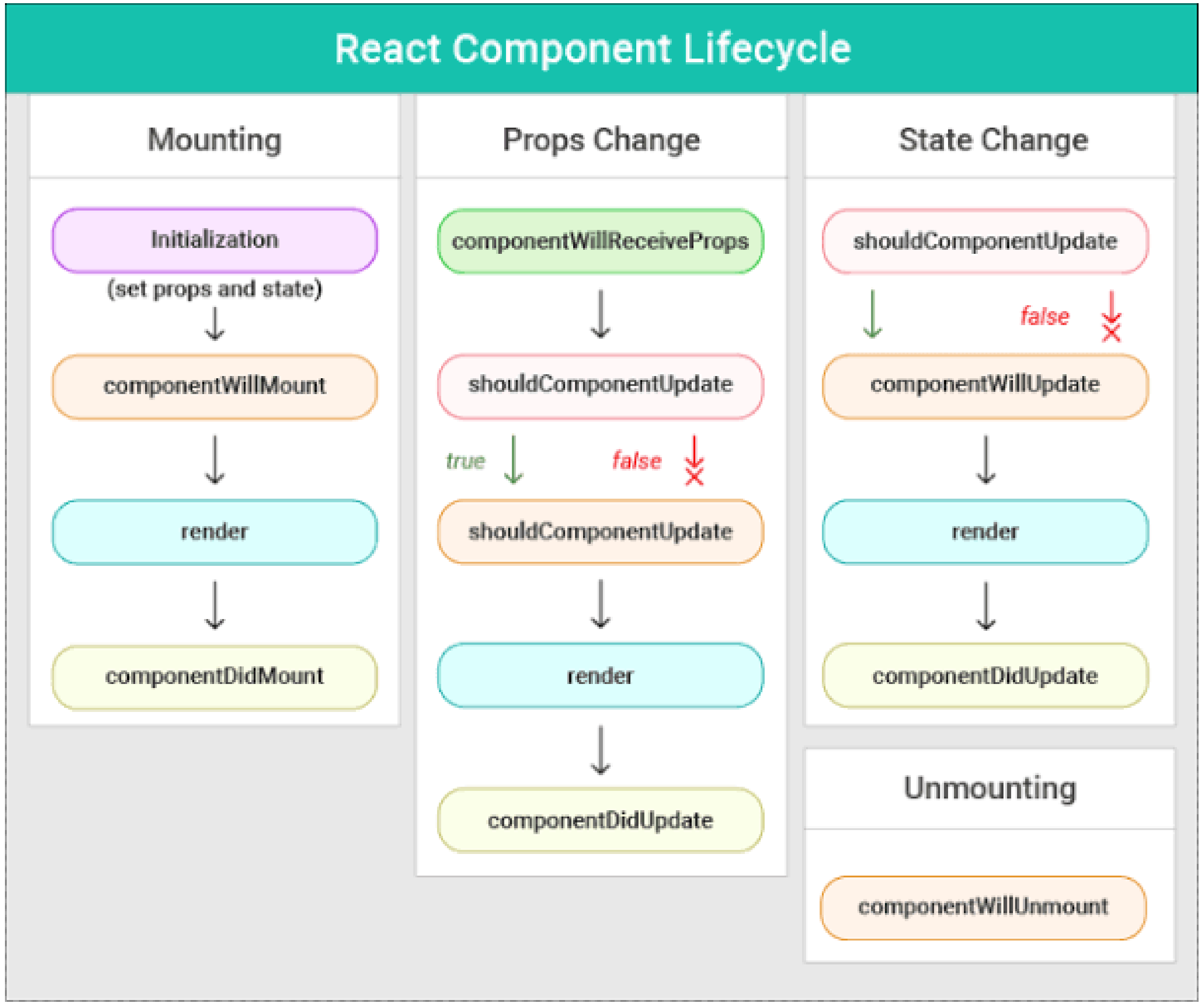
REACT COMPONENT LIFECYCLE – PROPS CHANGE

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



REACT COMPONENT LIFECYCLE – STATE CHANGE

- Occur when state is changed (via this.setState(..)) except inside componentWillMount method
- shouldComponentUpdate returning false results in followed methods won't be triggered also



THE LIFECYCLE UNMOUNTING

- ComponentWillUnmount - Used to clean up data

WHY REACTJS?

- React.js is extremely efficient – Virtual DOM
- It makes writing JavasScript easier - React.js uses a special syntax called JSX
- It’s awesome for SEO - Server rendering
- Componentized UI is the future of web development, and you need to start doing it now