Modern web application development

w3schools.com

THE WORLD'S LARGEST WEB DEVELOPER SI

TUTORIALS REFERENCES EXAMPLES

HTML and CSS

Learn HTML
Learn CSS
Learn W3.CSS
Learn Colors
Learn Bootstrap
Learn Icons
Learn Graphics
Learn How To

JavaScript

Learn JavaScript
Learn jQuery
Learn AngularJS
Learn JSON
Learn AJAX
Learn W3.JS
Learn AppML

Server Side

Learn SQL
Learn PHP
Learn ASP
Learn Node.js

Web Building

Web Templates
Web Statistics
Web Certificates

XML Tutorials

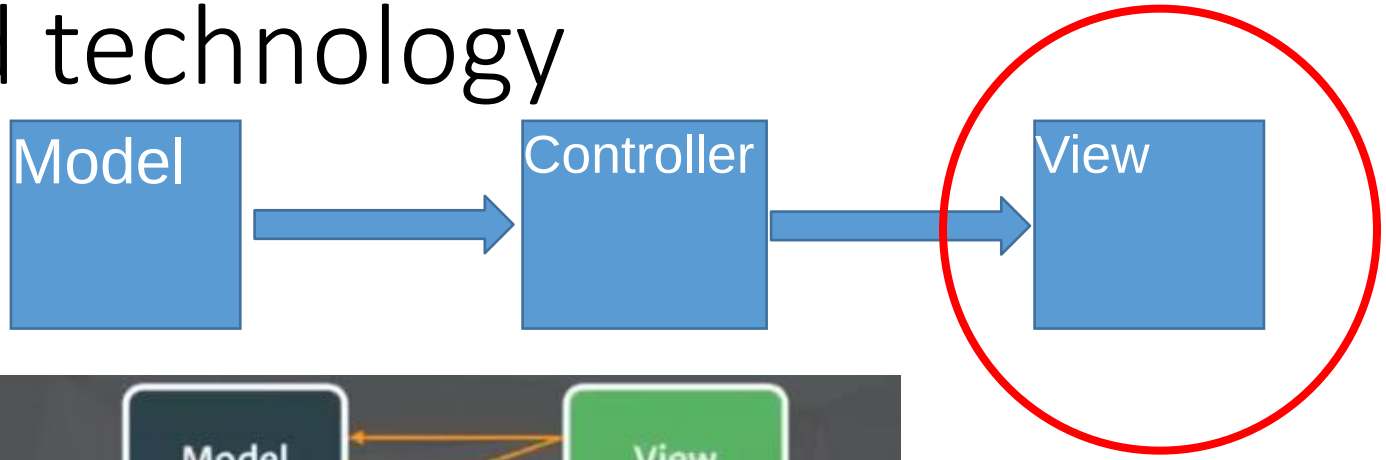
Learn XML
Learn XML AJAX
Learn XML DOM
Learn XML DTD
Learn XML Schema
Learn XSLT
Learn XPath
Learn XQuery

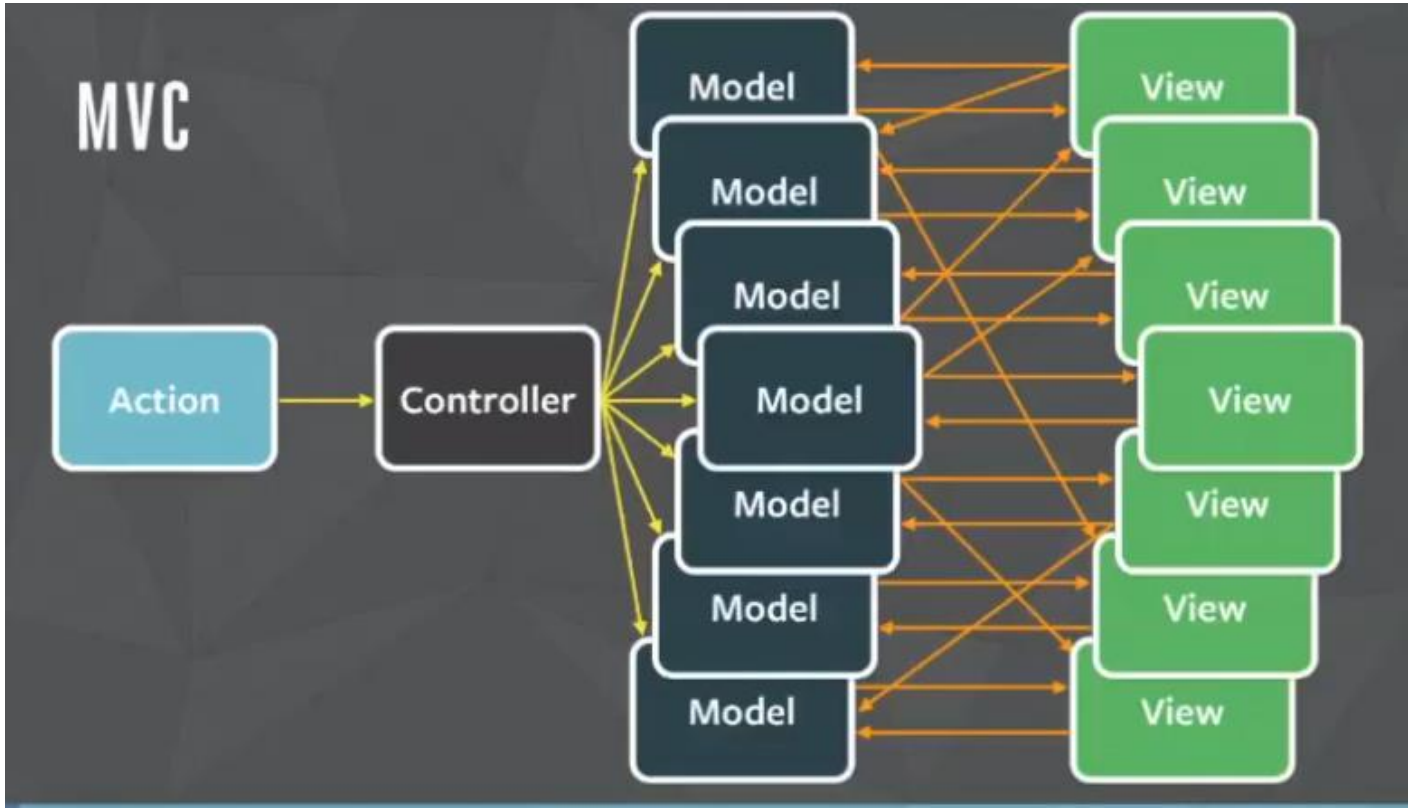
Web evolution

- Web has evolved from simple static pages to complex real time pages

Problems with old technology

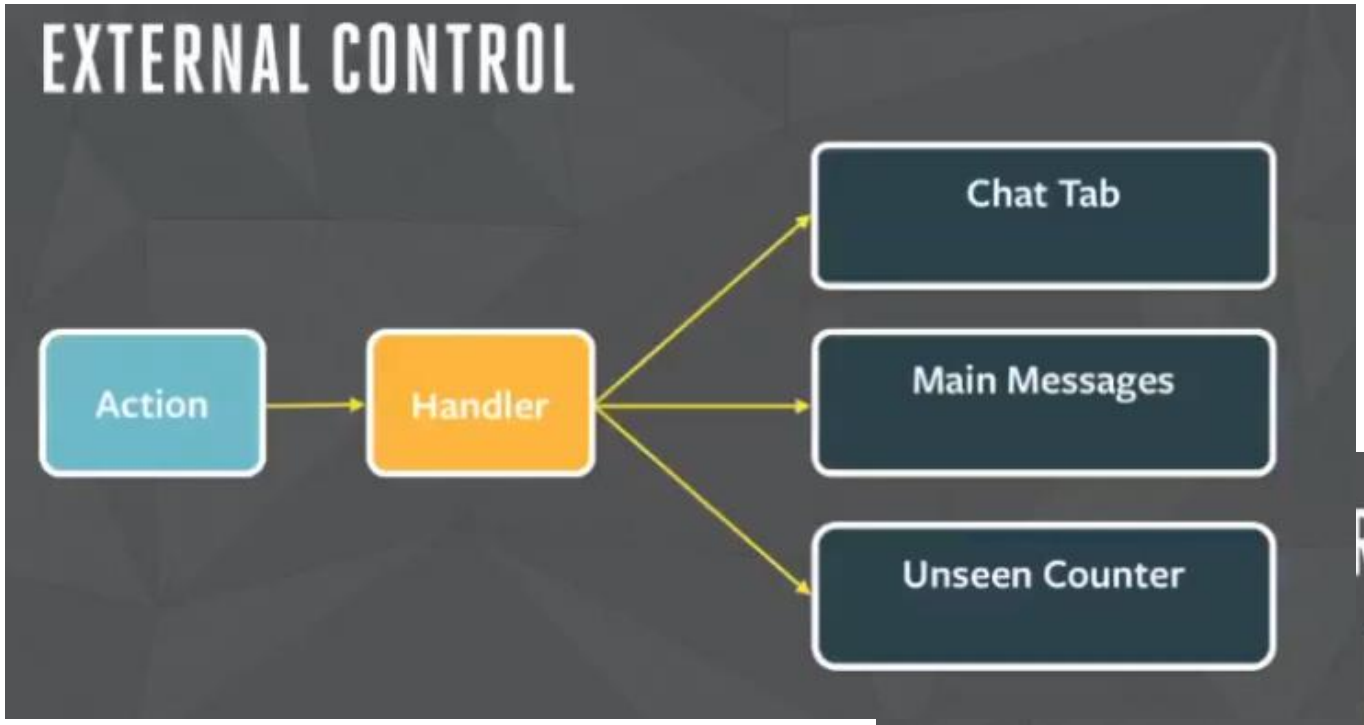
- MVC does not scale
- Typical MVC architecture

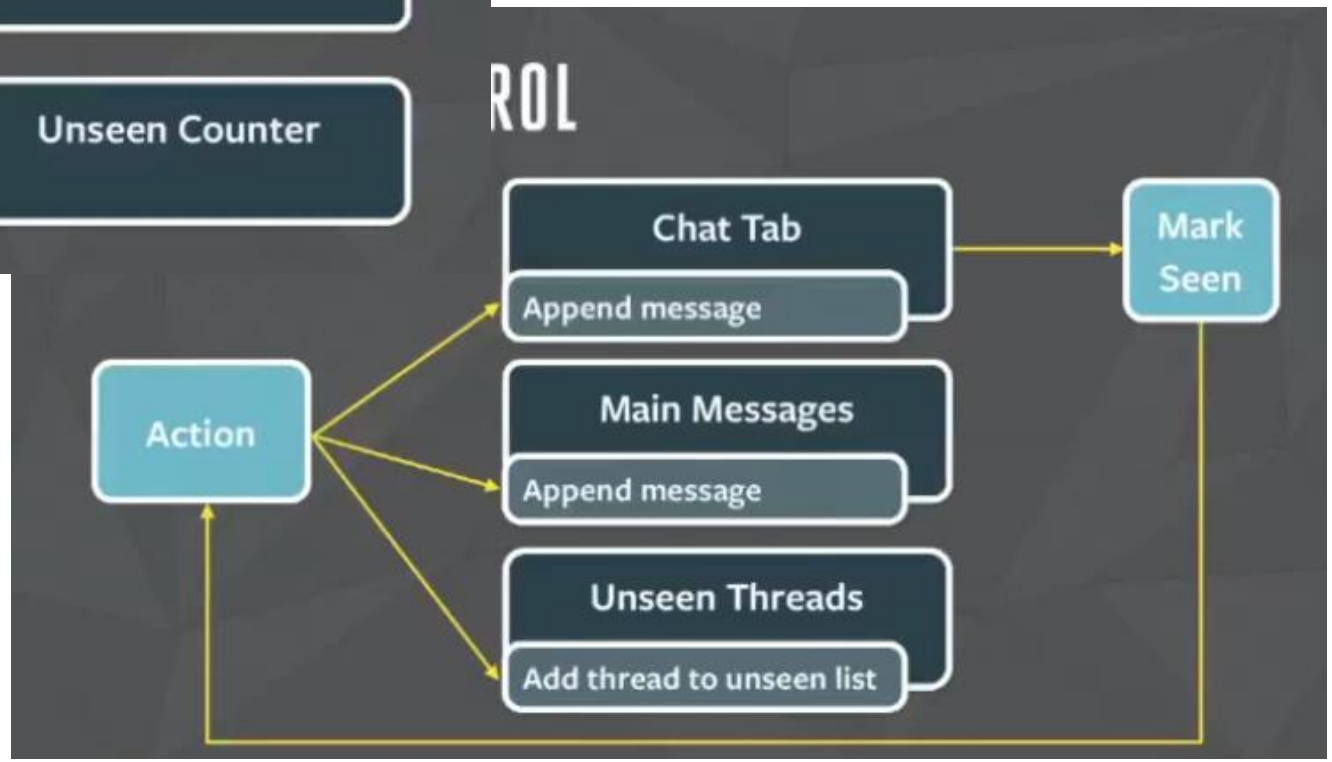




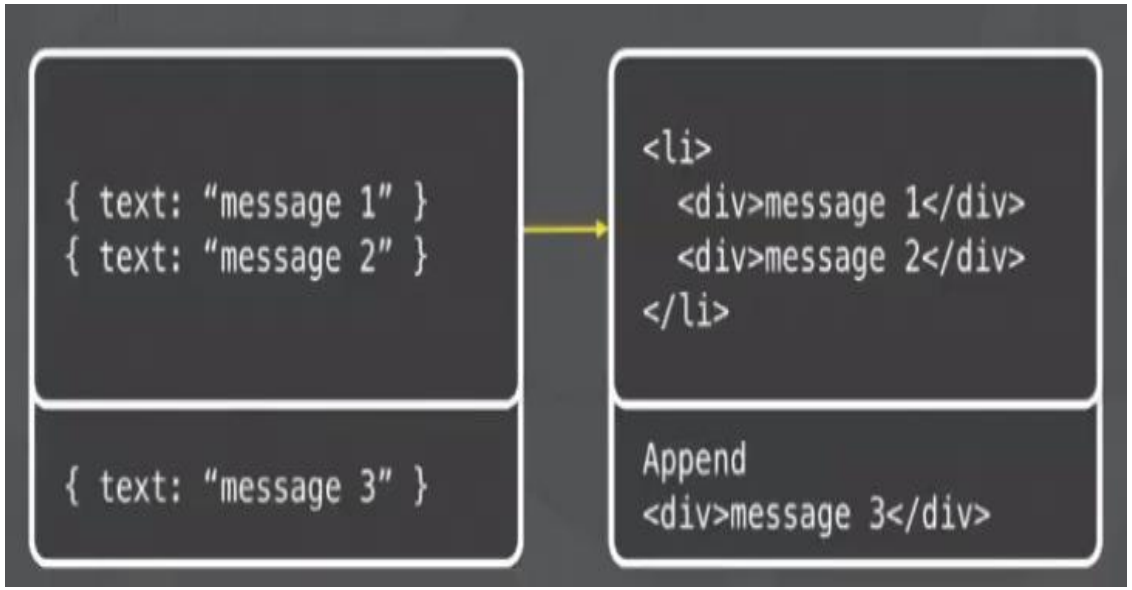
Problems with old technology

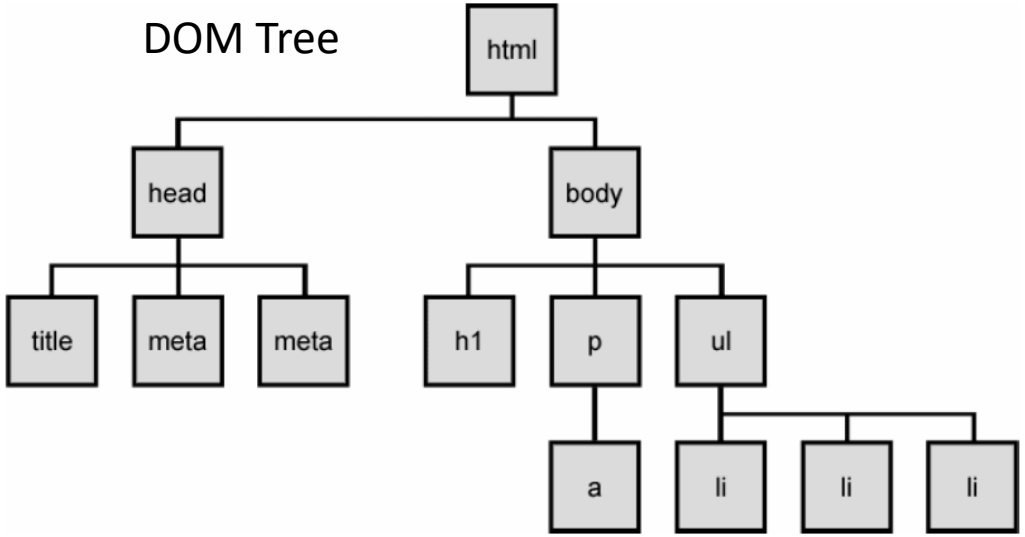
- Hard to test/manage state in a web component
- For simple chat system
  - Pull online users (GET /onlineusers) [admin, joe, john]
  - Count unread chat messages
  - If chat is closed don’t show online users
  - If more than 3 chat window is open show only last 3 opened chat window


EXTERNAL CONTROL
Action
Handler
Chat Tab
Main Messages
Unseen Counter


ROL
Action
Chat Tab
Append message
Main Messages
Append message
Unseen Threads
Add thread to unseen list
Mark Seen

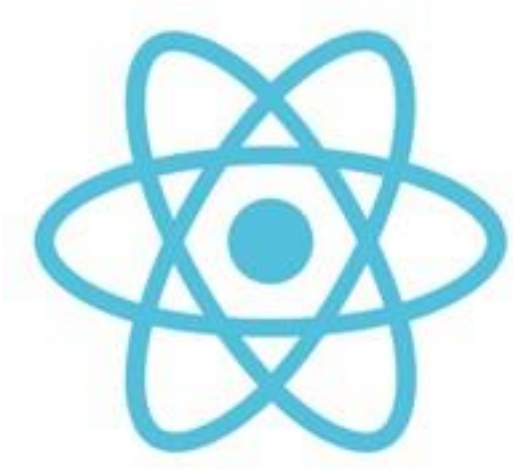
- HTML is designed for static page.
- It can run on almost any Javascript-enabled browser (React doesn't support IE < 9).
- DOM manipulation is expensive. (because of browser reflows)
- When data changes, refresh
- When data changes, React re-renders the component





Client side frameworks

- **ReactJS https://facebook.github.io/react/**
- Javascript library for building user interfaces
- Developed and maintained by facebook
- **Key Features**
  - JSX
  - Components
  - Unidirectional data flow



Most Forked Repos (Click to View Repo Link on GitHub)

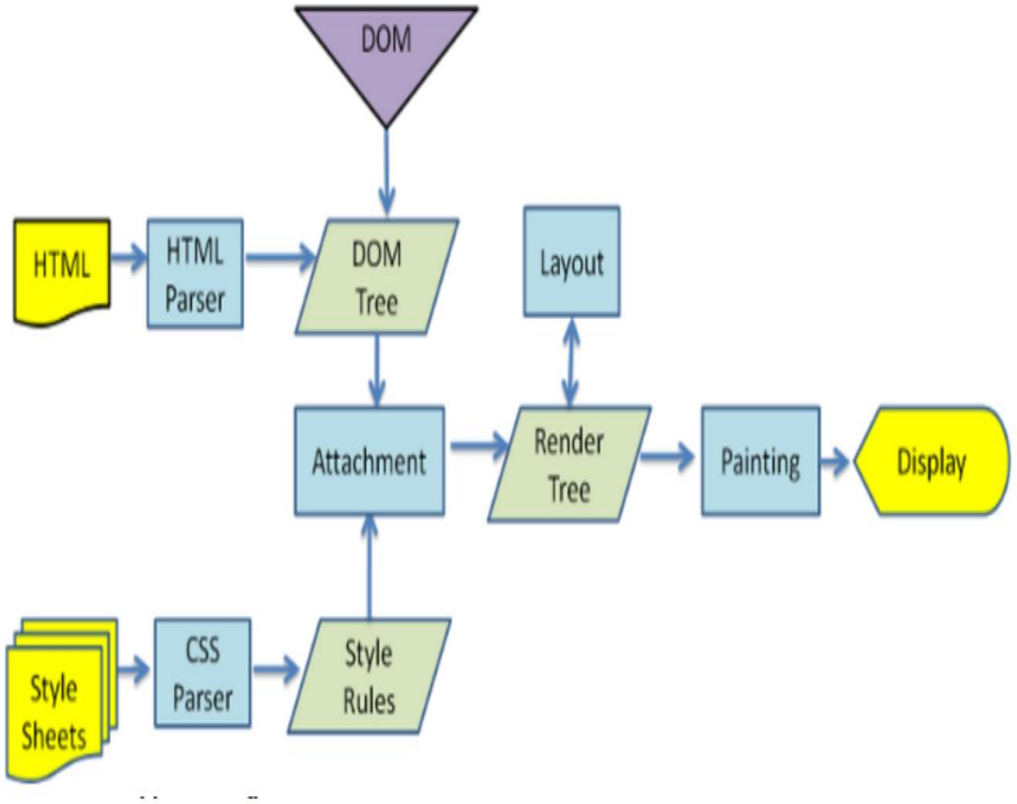
2015

tensorflow/tensorflow	Open source software library for numerical computation using data flow graphs.	4,355	1
facebook/react-native	A framework for building native apps with React.	4,198	2
NARKOZ/hacker-scripts	Based on a true story	3,553	3
apple/swift	The Swift Programming Language	3,068	4

Update an element in a DOM tree

- Update an element in DOM tree

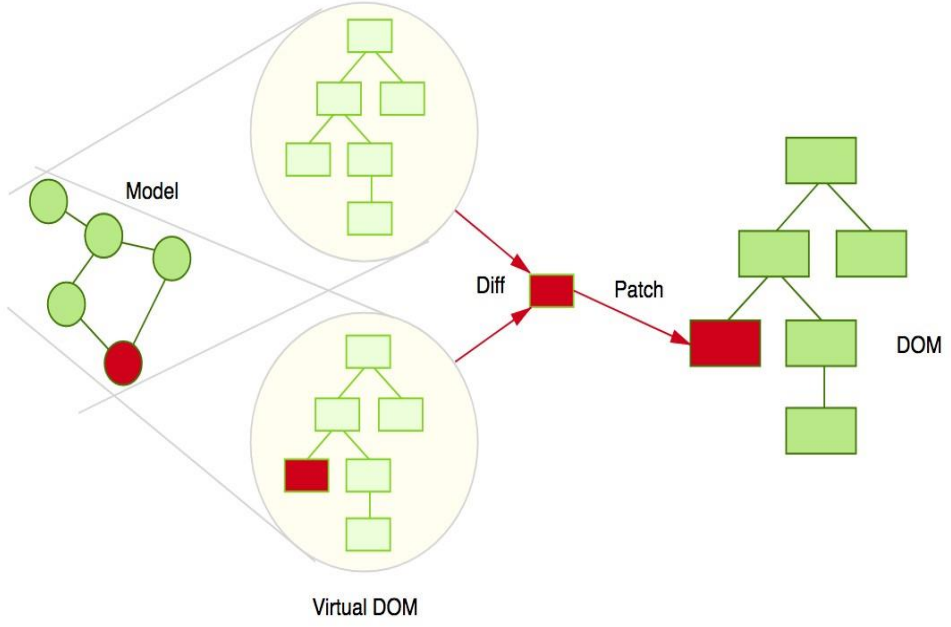
1. Browser have to parses the HTML
2. It removes the child element of elementId
3. Updates the DOM with the “New Value”
4. Re-calculate the CSS for the parent and child
5. Update the layout i.e. each elements exact co-ordinates on the screen
6. Traverse the render tree and paint it on the browser display



ReactJS

- Virtual DOM. Keeping state of DOM is hard
- Efficient diff algorithm
- Batched update operations
- Efficient update of sub tree only
- Uses observable instead of dirty checking to detect change

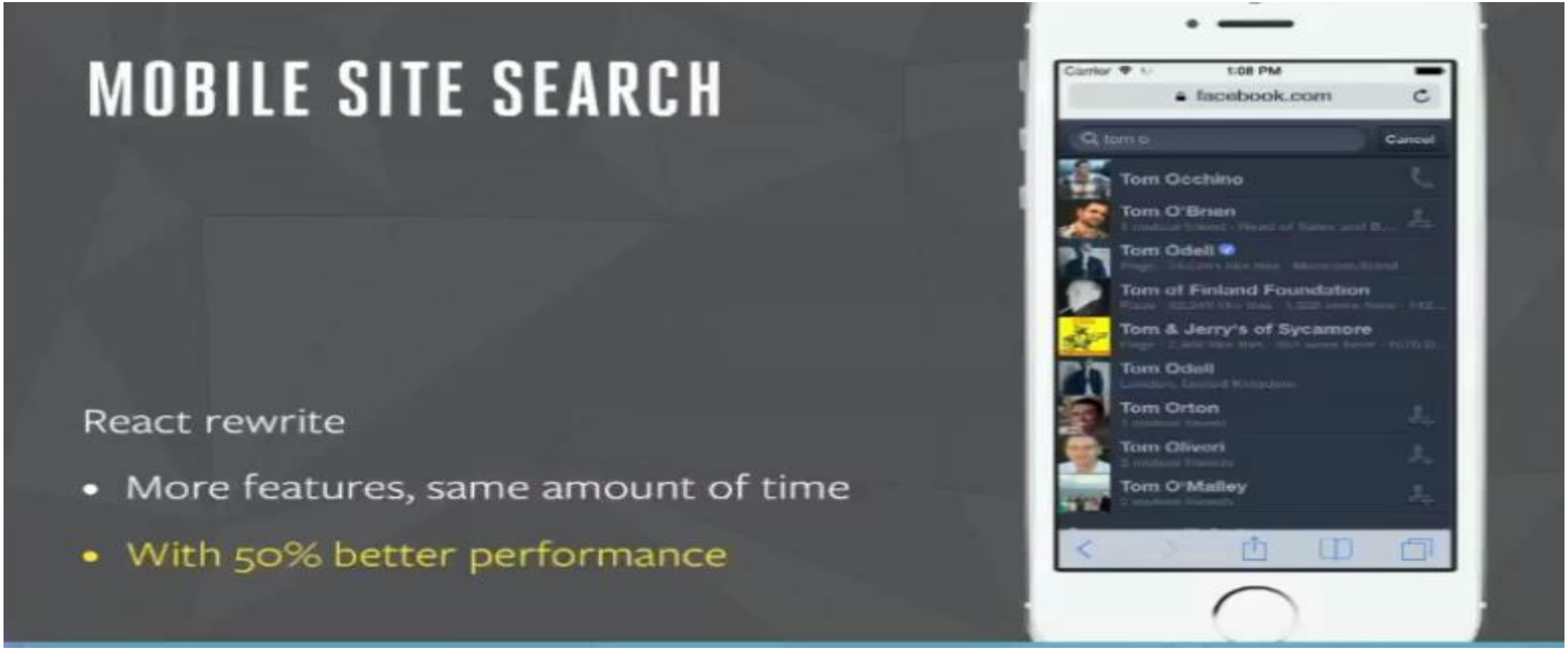
- AngularJS uses dirty checking runs in cycle after a specified time checking the whole model reduces the performance and thus makes the application slow.



Virtual DOM

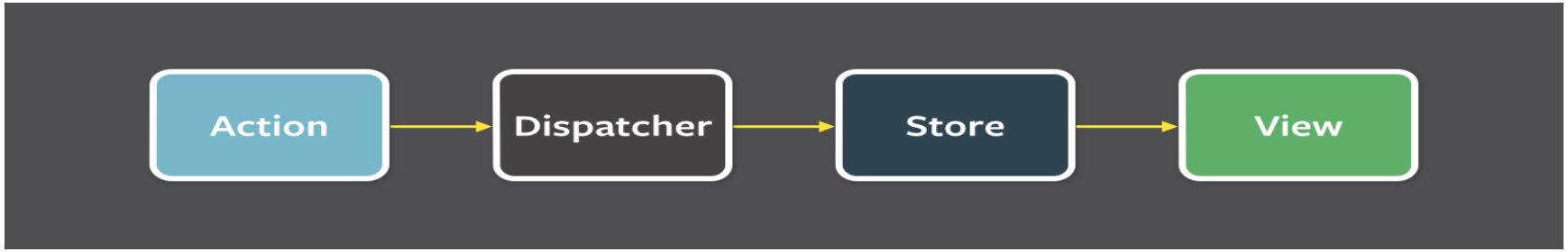
- When data changes, React re-renders the component
- Re-render all the children if parent state has changed.
- Breadth First Search.
- Reconciliation. (https://facebook.github.io/react/docs/reconciliation.html)
- Batch Update
  - ReactJS using the diff algorithm to find the minimum number of steps to update the Real DOM

React: 50% better performance



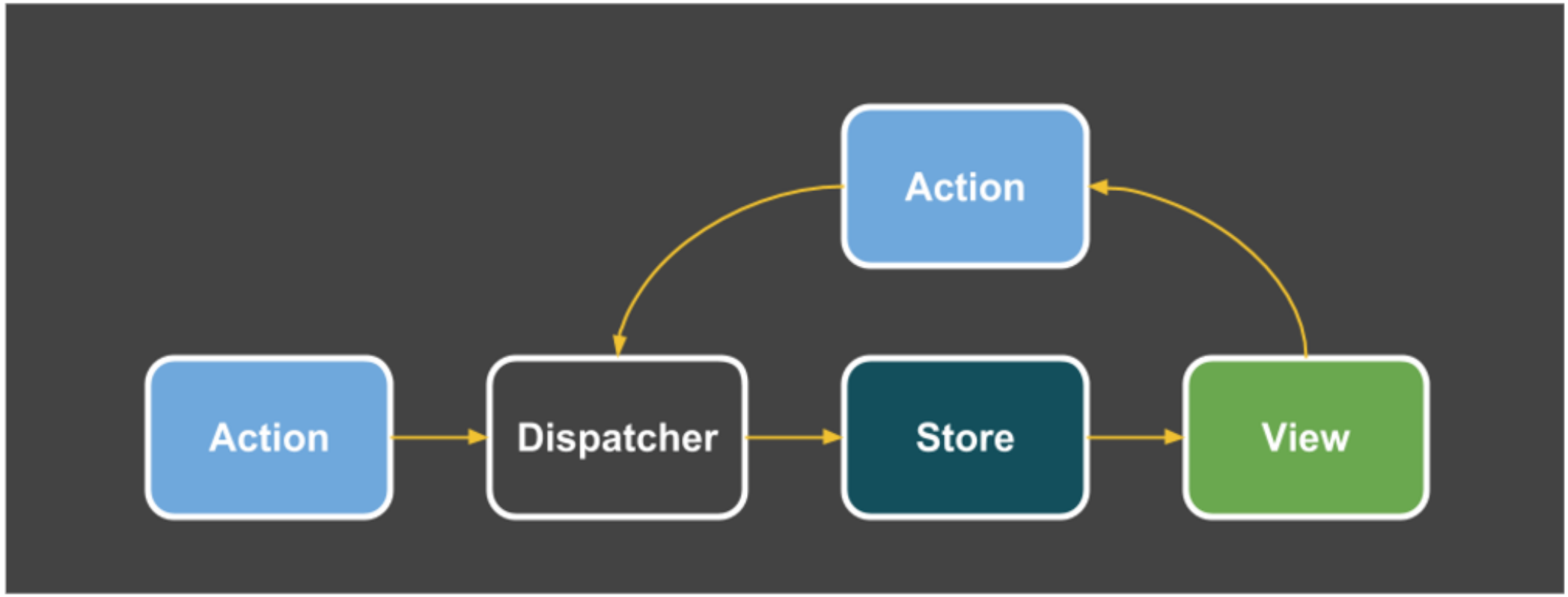
Two way vs. Unidirectional data flow

- In two way data binding
  - the view is updated when the state changes, and vice versa.
  - For example, when you change a model in AngularJS the view automatically reflects the changes.
  - it can lead to cascading updates and changing one model may trigger more updates.
  - View ⟷ State
- Unidirectional data flow
  - Mutation of data is done via actions. So, new data always enters into the store through actions.
  - View components subscribe to the stores and automatically re-render themselves using the new data. So, the data flow looks like this :



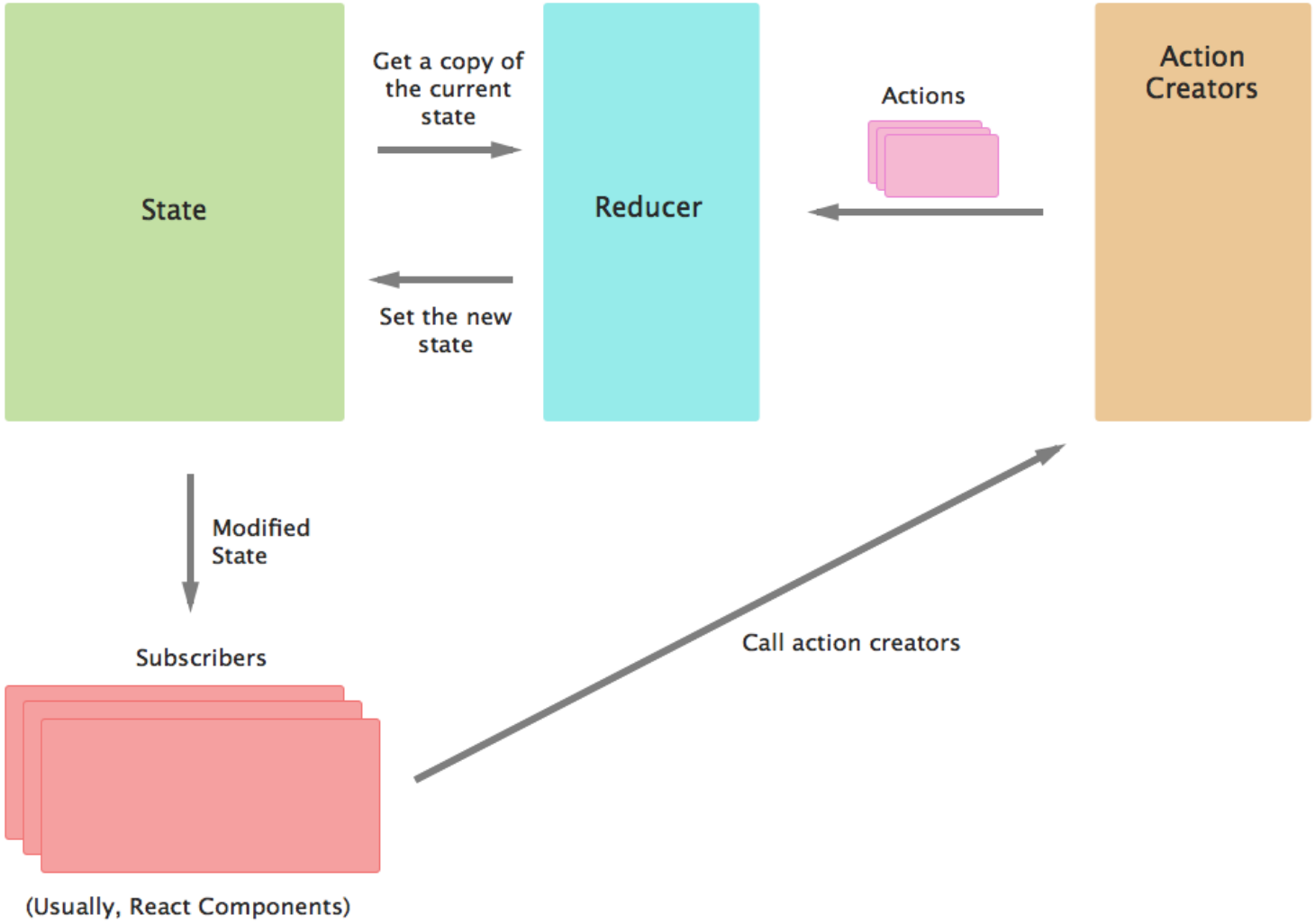
Redux

- Redux is a predictable state container for JavaScript apps.

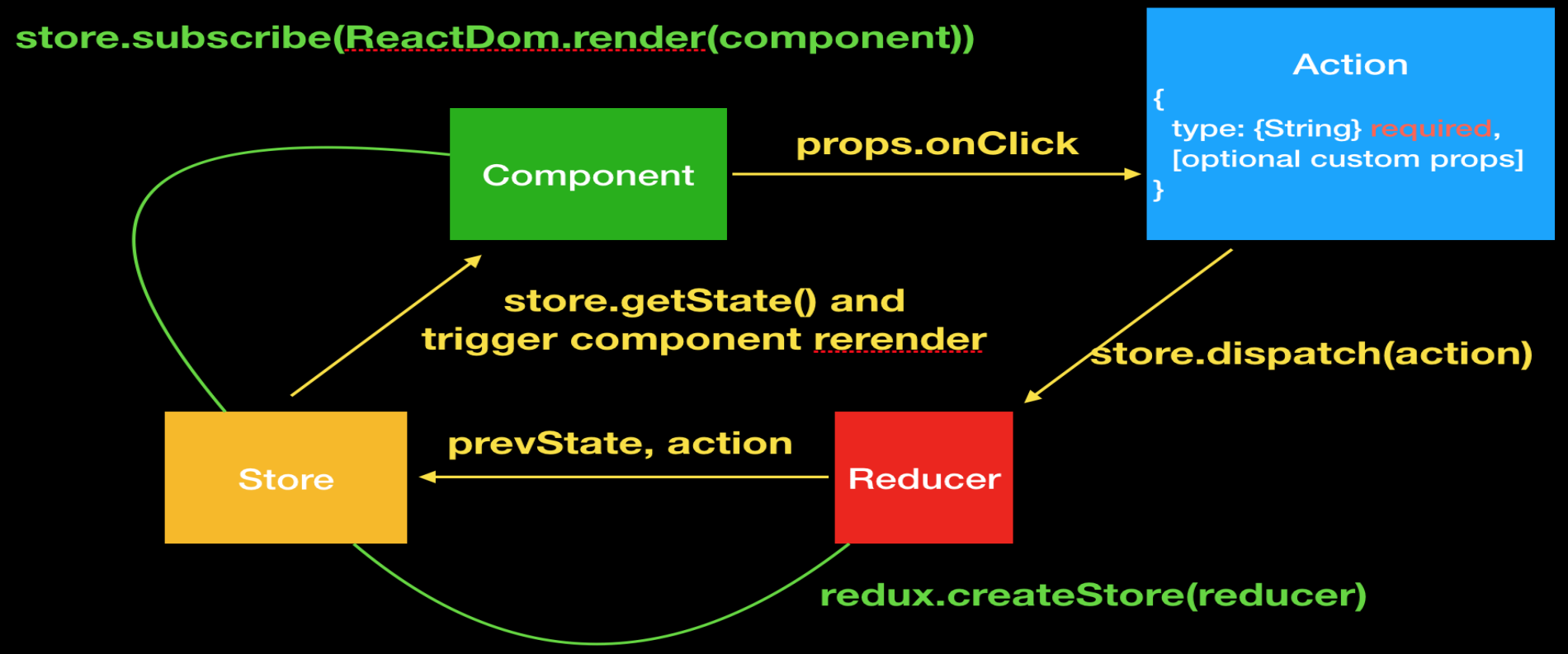


Redux

- Action Creator
- Reducers
- Store
-



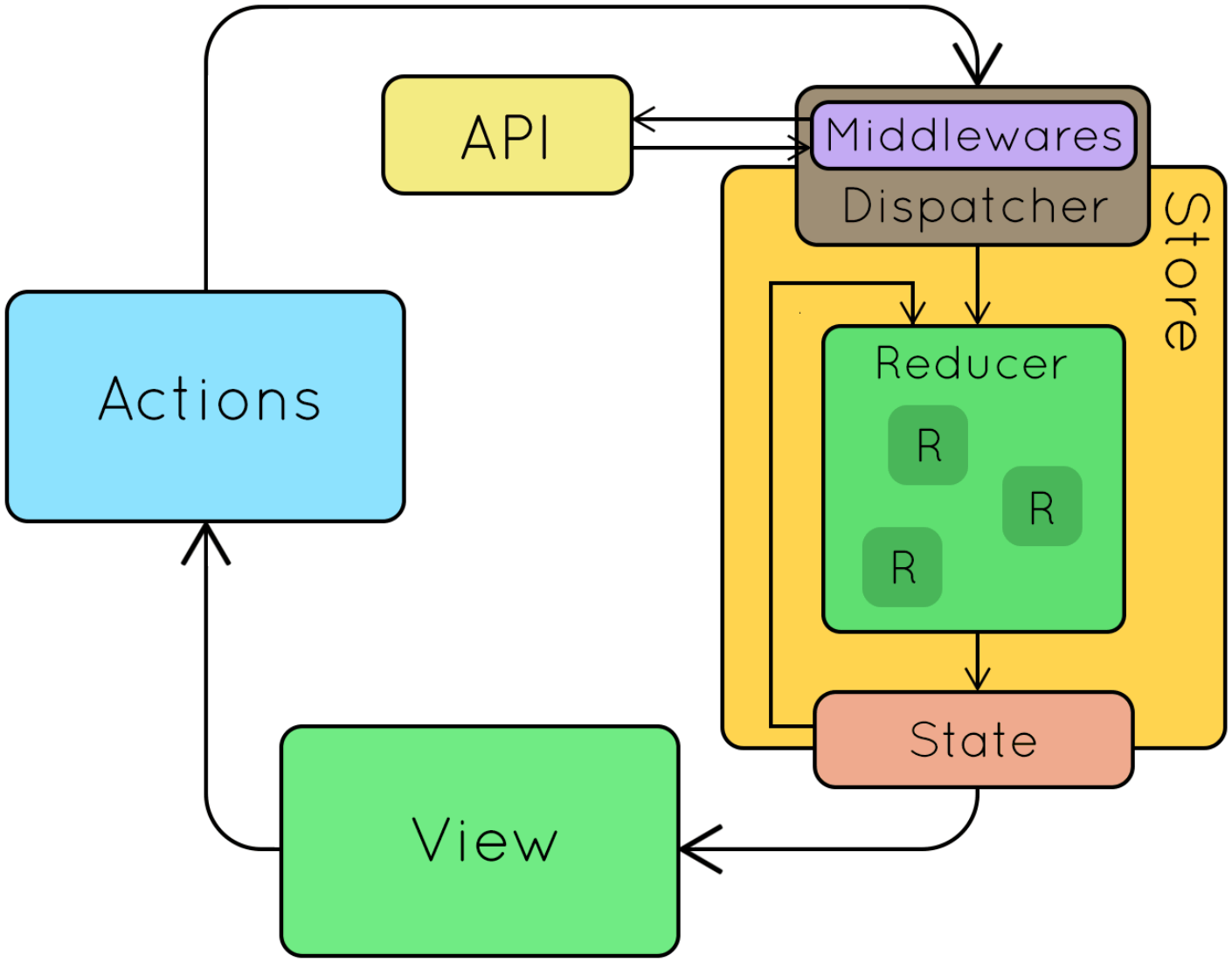
Redux Data Flow



Note:

- prevState: previous state in store before reducer executed
- Action [optional custom props] is used to transfer data
- renrender component transfers the new state from Store to component as props

green line and text means connection
yellow line and text means data flow


API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View

ReactJS

- Components (JSX)
- Javascript syntax extension, looks similar to XML

```
class HelloMessage extends React.Component {
  render() {
    return <div>Hello {this.props.name}</div>;
  }
}

ReactDOM.render(<HelloMessage name="Jane" />, mountNode);
```

https://facebook.github.io/react/tutorial/tutorial.html

React Docs Tutorial Community Blog Search docs... GitHub v15.6.1

TUTORIAL

Tutorial: Intro To React

Edit on GitHub

Before We Start

What We're Building

Today, we're going to build an interactive tic-tac-toe game.

If you like, you can check out the final result here: Final Result. Don't worry if the code doesn't make sense to you yet, or if it uses an unfamiliar syntax. We will be learning how to build this game step by step throughout this tutorial.

Try playing the game. You can also click on a link in the move list to go "back in time" and see what the board looked like just after that move was made.

Once you get a little familiar with the game, feel free to close that tab, as we'll start from a simpler template in the next sections.

Prerequisites

We'll assume some familiarity with HTML and JavaScript but you should be able to follow

Package Manager

- Advantages of Yarn
  - Speed – Better caching, pull dependencies in parallel
  - Deterministic – lock all package version and its dependencies

Web Server Side Technology

- Expressjs
  - Web application framework for nodejs

```
const express = require('express' 4.15.3 )
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.listen(3000, function () {
  console.log('Example app listening on port 3000!')
})
```

Composition of Components

- Components

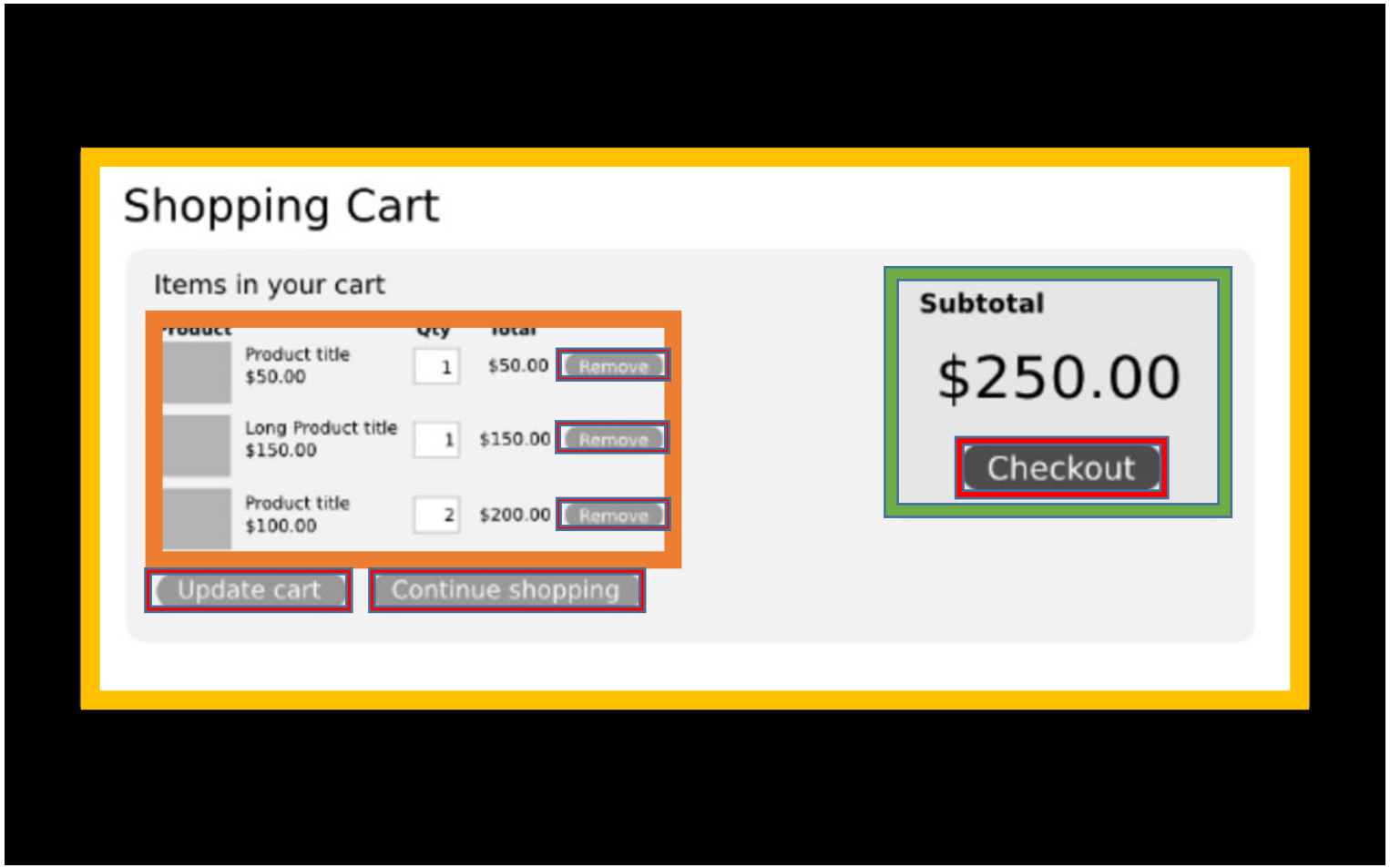
```
<Page />
<Article />
<Sidebar />
```

- Components

```
<Page>
  <Article />
  <Sidebar />
</Page>
```

Research:
https://www.linkedin.com/pulse/compose-me-function-composition-javascript-kevin-greene/

Composition of Components



Hello!

```
<html>
…
<body>
<div id=”container”></div>
</body>
</html>
```

```
ReactDOM.render(
  <div>Hello!</div>,
  document.getElementById('container')
);
```

```
ReactDOM.render(
   element,
   container,
   [callback] )
```

Component and JSX

- JSX is a preprocessor step that adds XML syntax to JavaScript
  - Not mandatory to use with React
  - Makes React more elegant
- JSX Code

```
class HelloMessage extends React.Component {
    render() {
        return <div>Hello {this.props.name}</div>;
    }
}
ReactDOM.render(<HelloMessage name="John" />, mountNode);
```

- Compiled JS

```
class HelloMessage extends React.Component {
    render() {
        return React.createElement( "div", null, "Hello ", this.props.name );
    }
}
ReactDOM.render(React.createElement(HelloMessage, { name: "John" }), mountNode);
```

State

- Components manage their own state
- React updates the page when state changes
- State can be passed to child via *props*

```
class User extends React.Component {
    state = {
        username: 'John'
    }

    render(){
        return(
            <p> Username: {this.state.username} </p>
        )
    }
}
```

Research:
https://facebook.github.io/react/docs/thinking-in-react.html#step-4-identify-where-your-state-should-live

Component Lifecycle Events

- **Mounting -** These methods are called when an instance of a component is being created and inserted into the DOM:
  - constructor()
  - componentWillMount()
  - render()
  - componentDidMount() ← Use this for loading remote data
- **Updating -** An update can be caused by changes to props or state. These methods are called when a component is being re-rendered:
  - componentWillReceiveProps()
  - shouldComponentUpdate()
  - componentWillUpdate()
  - render()
  - componentDidUpdate()
- **Unmounting -** This method is called when a component is being removed from the DOM:
  - componentWillUnmount()

Research:
https://facebook.github.io/react/docs/react-component.html#the-component-lifecycle

Homework

Food Ordering App