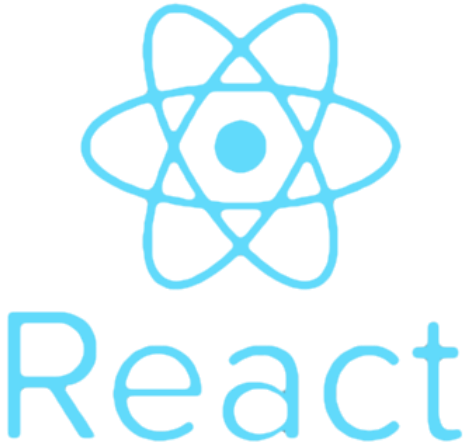
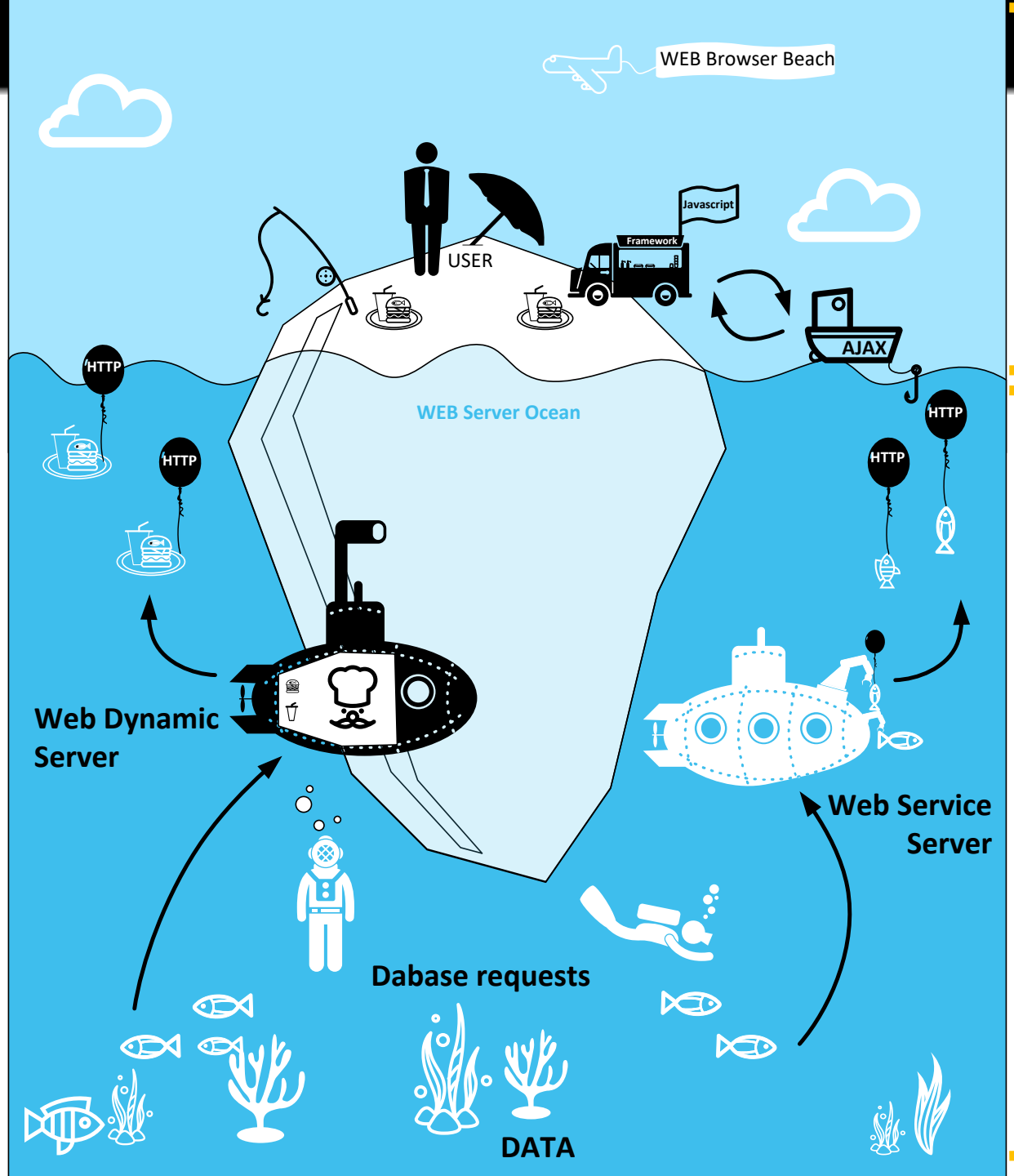
Introduction to React.js (and Redux!)



CPE LYON
ÉCOLE SUPÉRIEURE DE CHIMIE PHYSIQUE ÉLECTRONIQUE DE LYON

What is a Front-End Framework ?

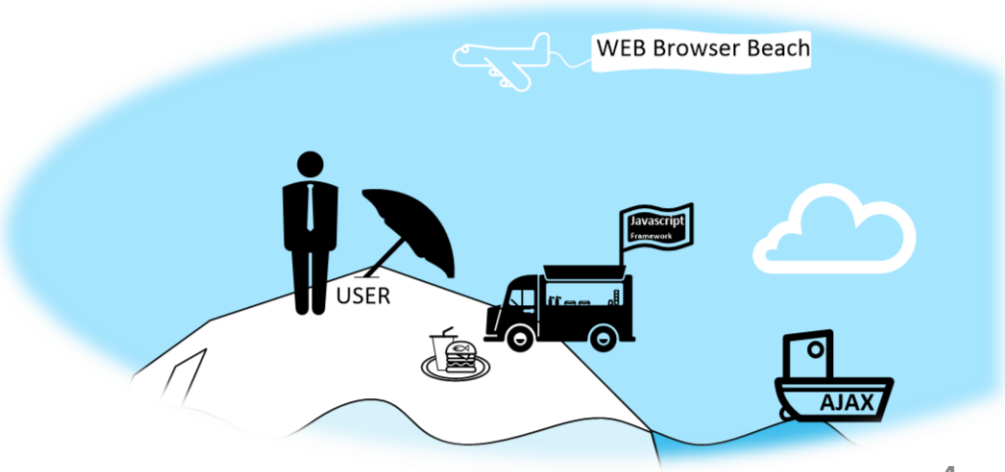


FRONT END

BACK END

Front End

- Everything running on the web browser !!
- Basic languages
  - HTML
  - CSS
  - JAVASCRIPT
- Lots of toolboxes !!
  - Jquery
  - AJAX
  - Canvas
  - WebGL
  - …



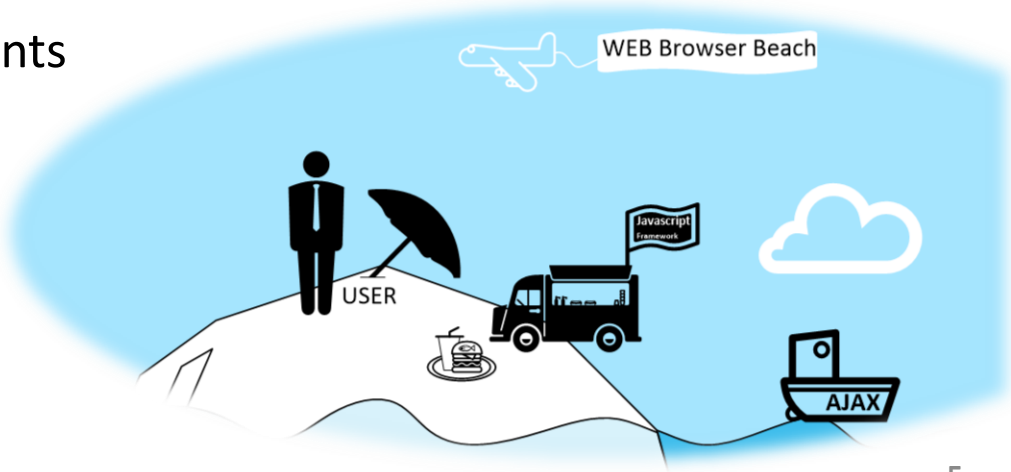
Why do we need additional tools ?

- Everything can be done through pure JAVASCRIPT !

  → right but hard!

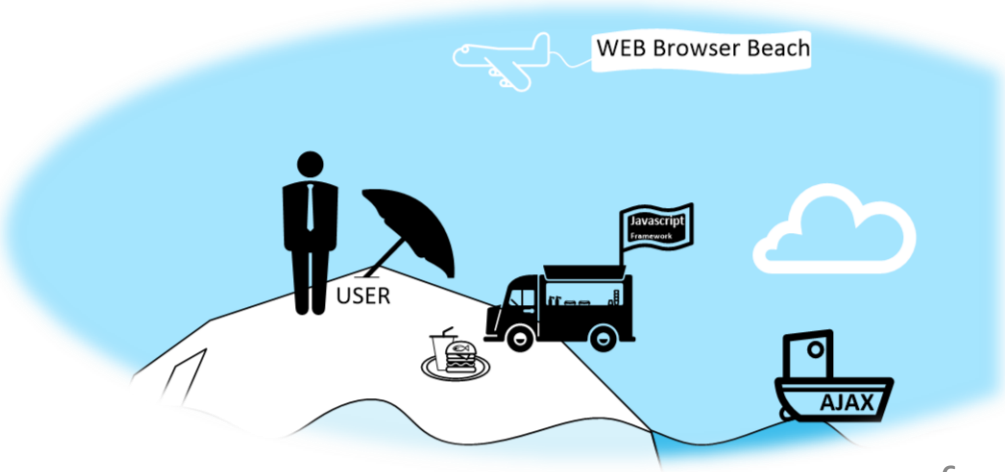
- Front End Framework
  - Help to organize front end development
  - Provide lots of predefined components
  - Allow the creation of components
  - Help to gain time !!

(depending on your front end framework knowledge !)



What is the best Frond End Framework ?

- It depends of what you want!
  - Time to learn
  - Front End efficiency
  - Modularity
  - Component creation complexity
  - Community Size
  - Maturity



Frond-End Solutions Comparison

	AngularJS	Angular 2	ReactJS	Vue.js
Definition	MVW framework	MVC framework	JavaScript library	MVC framework
1st Release	2009	2016	2013	2014
Homepage	angularjs.org	angular.io	reactjs.net	vuejs.org
# Contributors on GitHub	1,562	392	912	62
GitHub Star Rating	54,402	19,832	57,878	39,933

https://www.valuecoders.com/blog/wp-content/uploads/2016/11/Angular-react-and-vue-comparision.png

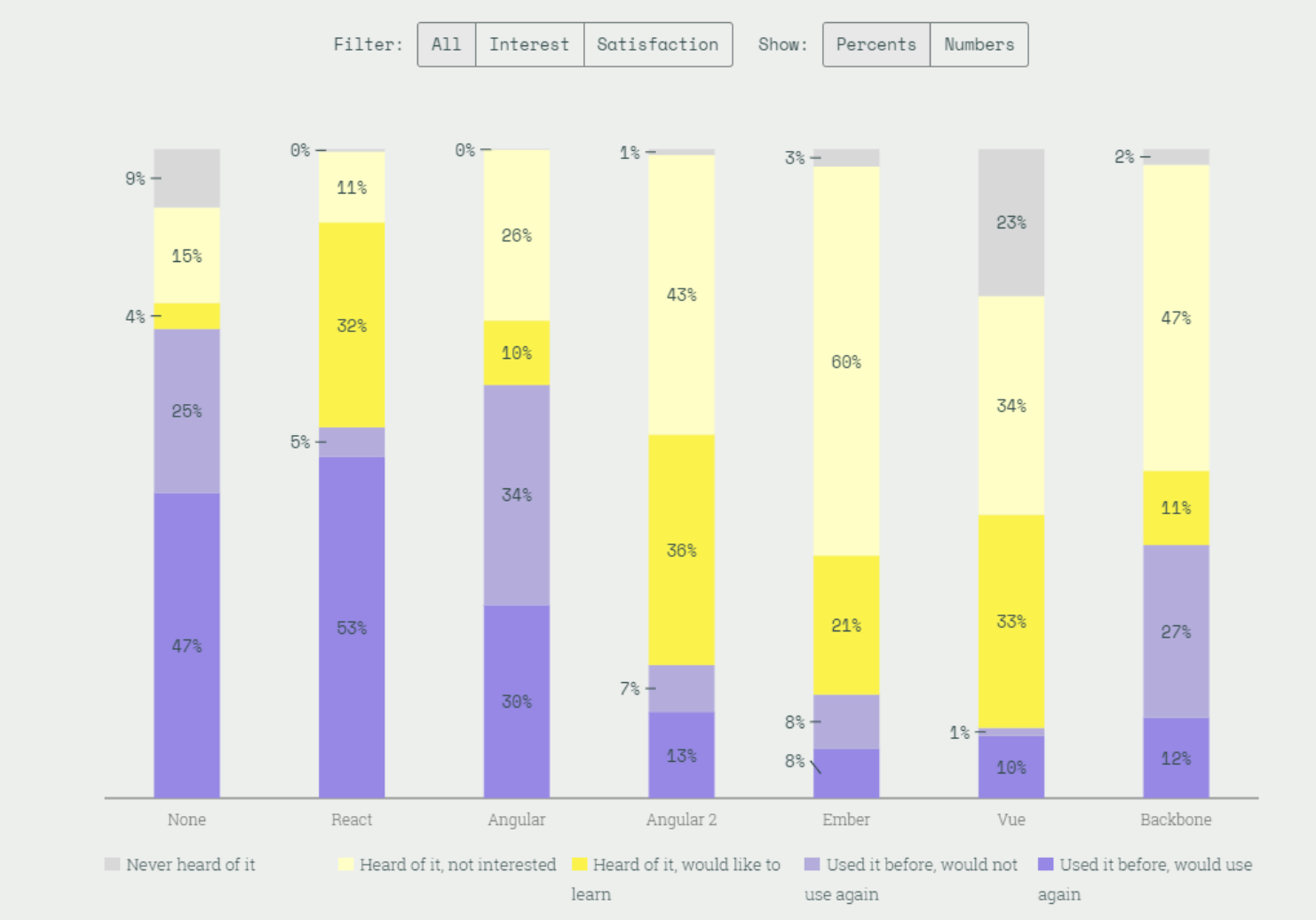
Frond-End Solutions Comparison

Duration in milliseconds (Slowdown = Duration / Fastest)

	angular v1.5.8	angular v2.0.0-rc5	aurelia v1.0.0	bobril v4.44.1	cyclejs v7.0.0	domvm v1.2.10	inferno v0.7.26	inferno v1.0.0-alpha7	kivi v1.0.0-rc0	mithril v0.2.5	mithril v1.0.0-alpha	plastiq v1.33.0	preact v6.0.2	ractive-edge	ractive v0.7.3	react-lite v0.15.17	react v15.3.1	react v15.3.1-mobX-v2.5.0	riot v2.6.1	tsers v1.0.0	vidom v0.3.18	vue v1.0.26	vue v2.0.0-beta1	vanillajs
create rows Duration for creating 1000 rows after the page loaded.	270.40 ± 12.51 (2.31)	198.06 ± 7.91 (1.69)	174.16 ± 3.44 (1.49)	139.71 ± 3.08 (1.19)	158.15 ± 5.26 (1.35)	170.28 ± 8.35 (1.45)	149.64 ± 2.03 (1.28)	136.06 ± 2.37 (1.16)	128.53 ± 1.30 (1.10)	250.09 ± 18.30 (2.13)	152.84 ± 3.65 (1.30)	172.47 ± 7.74 (1.47)	182.90 ± 4.48 (1.56)	350.95 ± 17.65 (2.99)	443.95 ± 21.97 (3.79)	171.30 ± 10.08 (1.46)	187.28 ± 8.94 (1.60)	227.44 ± 6.55 (1.94)	385.41 ± 12.81 (3.29)	266.09 ± 18.28 (2.27)	145.51 ± 4.00 (1.24)	225.90 ± 16.30 (1.93)	171.36 ± 5.15 (1.46)	117.26 ± 8.95 (1.00)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	235.25 ± 16.15 (4.38)	178.45 ± 1.98 (3.32)	81.52 ± 2.16 (1.52)	153.88 ± 2.46 (2.87)	71.39 ± 0.71 (1.33)	73.85 ± 4.97 (1.38)	157.64 ± 2.68 (2.94)	53.69 ± 1.22 (1.00)	131.39 ± 6.03 (2.45)	229.39 ± 2.18 (4.27)	275.39 ± 8.25 (5.13)	174.25 ± 3.42 (3.25)	190.28 ± 2.00 (3.54)	65.35 ± 0.73 (1.22)	76.55 ± 1.76 (1.43)	209.67 ± 2.45 (3.91)	190.16 ± 2.20 (3.54)	211.71 ± 2.99 (3.94)	74.66 ± 1.59 (1.39)	116.17 ± 1.78 (2.16)	159.10 ± 5.19 (2.96)	227.03 ± 12.75 (4.23)	68.76 ± 0.93 (1.28)	54.17 ± 1.88 (1.01)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	14.32 ± 1.06 (1.00)	11.42 ± 1.23 (1.00)	13.87 ± 2.96 (1.00)	11.59 ± 0.57 (1.00)	28.53 ± 1.25 (1.78)	28.68 ± 3.20 (1.79)	16.10 ± 2.56 (1.01)	12.95 ± 1.72 (1.00)	12.22 ± 1.42 (1.00)	55.79 ± 1.36 (3.49)	18.55 ± 1.35 (1.16)	23.03 ± 3.13 (1.44)	13.73 ± 0.98 (1.00)	15.14 ± 2.95 (1.00)	31.23 ± 1.84 (1.95)	28.98 ± 1.05 (1.81)	16.40 ± 1.07 (1.03)	14.77 ± 1.97 (1.00)	28.98 ± 0.83 (1.81)	35.24 ± 1.36 (2.20)	16.66 ± 3.24 (1.04)	15.46 ± 0.71 (1.00)	22.17 ± 0.74 (1.39)	12.09 ± 2.89 (1.00)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	4.34 ± 1.34 (1.00)	2.39 ± 0.24 (1.00)	11.47 ± 0.39 (1.00)	4.95 ± 2.16 (1.00)	22.34 ± 3.79 (1.40)	21.92 ± 2.51 (1.37)	5.67 ± 1.36 (1.00)	2.75 ± 0.32 (1.00)	3.02 ± 1.10 (1.00)	40.40 ± 1.44 (2.53)	9.50 ± 0.31 (1.00)	4.64 ± 0.68 (1.00)	5.01 ± 1.44 (1.00)	7.26 ± 0.75 (1.00)	9.35 ± 0.32 (1.00)	20.26 ± 1.01 (1.27)	5.96 ± 0.64 (1.00)	4.06 ± 0.46 (1.00)	19.84 ± 0.31 (1.24)	24.28 ± 0.80 (1.52)	6.37 ± 2.21 (1.00)	7.27 ± 1.33 (1.00)	13.30 ± 1.06 (1.00)	3.17 ± 1.26 (1.00)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	50.09 ± 1.08 (3.13)	50.16 ± 2.81 (3.13)	51.15 ± 2.82 (3.20)	45.51 ± 2.53 (2.84)	23.82 ± 0.86 (1.49)	29.80 ± 1.43 (1.86)	50.62 ± 2.44 (3.16)	13.20 ± 2.63 (1.00)	47.49 ± 2.72 (2.97)	94.47 ± 2.17 (5.90)	55.00 ± 1.32 (3.44)	47.75 ± 2.41 (2.98)	49.33 ± 1.24 (3.08)	12.51 ± 2.79 (1.00)	29.37 ± 1.42 (1.84)	60.42 ± 0.26 (3.78)	48.25 ± 0.53 (3.02)	49.42 ± 2.76 (3.09)	26.38 ± 0.97 (1.65)	30.64 ± 0.63 (1.91)	53.22 ± 3.80 (3.33)	49.88 ± 1.67 (3.12)	19.14 ± 0.55 (1.20)	9.68 ± 1.67 (1.00)
remove row Duration to remove a row. (with 5 warmup iterations).	63.57 ± 0.67 (1.92)	64.11 ± 1.88 (1.94)	91.30 ± 1.14 (2.76)	62.56 ± 2.09 (1.89)	52.92 ± 3.85 (1.60)	50.48 ± 1.76 (1.53)	61.98 ± 1.92 (1.87)	33.82 ± 1.06 (1.02)	61.95 ± 1.36 (1.87)	96.53 ± 1.55 (2.92)	132.26 ± 1.59 (4.00)	68.43 ± 2.04 (2.07)	136.63 ± 0.63 (4.13)	75.31 ± 2.04 (2.28)	192.49 ± 1.84 (5.82)	78.45 ± 1.00 (2.37)	67.07 ± 2.54 (2.03)	72.57 ± 2.83 (2.19)	89.11 ± 0.80 (2.69)	72.03 ± 1.26 (2.18)	67.98 ± 1.46 (2.06)	66.26 ± 1.84 (2.00)	44.09 ± 0.77 (1.33)	33.07 ± 0.83 (1.00)
create many rows Duration to create 10,000 rows	2414.53 ± 92.56 (2.00)	1914.70 ± 63.86 (1.59)	1693.89 ± 17.11 (1.40)	1310.14 ± 23.91 (1.08)	1638.01 ± 17.57 (1.36)	1548.40 ± 28.56 (1.28)	1382.95 ± 18.20 (1.14)	1342.28 ± 10.63 (1.11)	1228.45 ± 25.49 (1.02)	2920.82 ± 119.22 (2.42)	1540.39 ± 22.98 (1.28)	1730.09 ± 14.18 (1.43)	2448.28 ± 98.62 (2.03)	3291.85 ± 132.68 (2.73)	5720.39 ± 39.30 (4.74)	2039.72 ± 47.09 (1.69)	1839.96 ± 29.79 (1.52)	2229.20 ± 81.40 (1.85)	4218.09 ± 75.67 (3.49)	2810.34 ± 13.33 (2.33)	1389.43 ± 7.91 (1.15)	2890.37 ± 313.40 (2.39)	1712.87 ± 8.13 (1.42)	1208.00 ± 11.56 (1.00)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	764.82 ± 39.92 (3.67)	594.38 ± 8.91 (2.86)	639.98 ± 11.15 (3.07)	222.20 ± 2.83 (1.07)	430.25 ± 2.52 (2.07)	506.71 ± 8.79 (2.43)	253.99 ± 6.92 (1.22)	211.87 ± 1.12 (1.02)	215.84 ± 6.50 (1.04)	1503.52 ± 29.27 (7.22)	355.81 ± 11.11 (1.71)	272.52 ± 8.00 (1.31)	384.51 ± 3.20 (1.85)	372.35 ± 8.14 (1.79)	1331.00 ± 24.46 (6.39)	1903.71 ± 59.04 (9.15)	297.09 ± 11.04 (1.43)	323.66 ± 12.47 (1.55)	4596.15 ± 504.29 (22.08)	626.72 ± 13.95 (3.01)	267.63 ± 14.23 (1.29)	694.65 ± 11.42 (3.34)	420.53 ± 8.23 (2.02)	208.15 ± 4.48 (1.00)
clear rows Duration to clear the table filled with 10.000 rows.	628.16 ± 38.98 (3.99)	281.60 ± 6.69 (1.79)	218.64 ± 5.19 (1.39)	185.20 ± 1.39 (1.18)	205.67 ± 2.22 (1.31)	276.12 ± 4.52 (1.76)	211.57 ± 3.05 (1.35)	198.38 ± 1.72 (1.26)	162.61 ± 1.84 (1.03)	245.22 ± 1.97 (1.56)	213.56 ± 2.22 (1.36)	205.39 ± 4.22 (1.31)	520.77 ± 13.82 (3.31)	573.03 ± 6.87 (3.64)	2256.17 ± 37.99 (14.35)	280.89 ± 5.52 (1.79)	371.16 ± 4.12 (2.36)	391.64 ± 5.79 (2.49)	862.28 ± 8.69 (5.48)	283.13 ± 8.43 (1.80)	174.44 ± 2.32 (1.11)	373.95 ± 8.32 (2.38)	223.87 ± 4.61 (1.42)	157.27 ± 1.93 (1.00)
clear rows a 2nd time Time to clear the table filled with 10.000 rows. But warmed up with only one iteration.	1338.26 ± 24.10 (8.50)	265.82 ± 2.22 (1.69)	204.41 ± 1.64 (1.30)	182.84 ± 1.76 (1.16)	207.96 ± 1.84 (1.32)	264.83 ± 7.20 (1.68)	205.70 ± 2.60 (1.31)	187.84 ± 2.41 (1.19)	160.89 ± 1.79 (1.02)	242.46 ± 3.16 (1.54)	204.96 ± 1.57 (1.30)	187.62 ± 1.99 (1.19)	488.38 ± 6.07 (3.10)	572.77 ± 8.39 (3.64)	1236.32 ± 68.61 (7.85)	303.16 ± 3.11 (1.92)	354.71 ± 7.91 (2.25)	372.56 ± 8.50 (2.36)	871.37 ± 10.77 (5.53)	290.06 ± 1.56 (1.84)	170.07 ± 1.71 (1.08)	368.67 ± 5.70 (2.34)	210.56 ± 4.41 (1.34)	157.53 ± 2.22 (1.00)
slowdown geometric mean	2.62	1.85	1.66	1.40	1.48	1.63	1.49	1.07	1.32	2.99	1.83	1.62	2.20	1.87	3.62	2.40	1.82	1.97	3.18	2.09	1.47	2.16	1.37	1.00

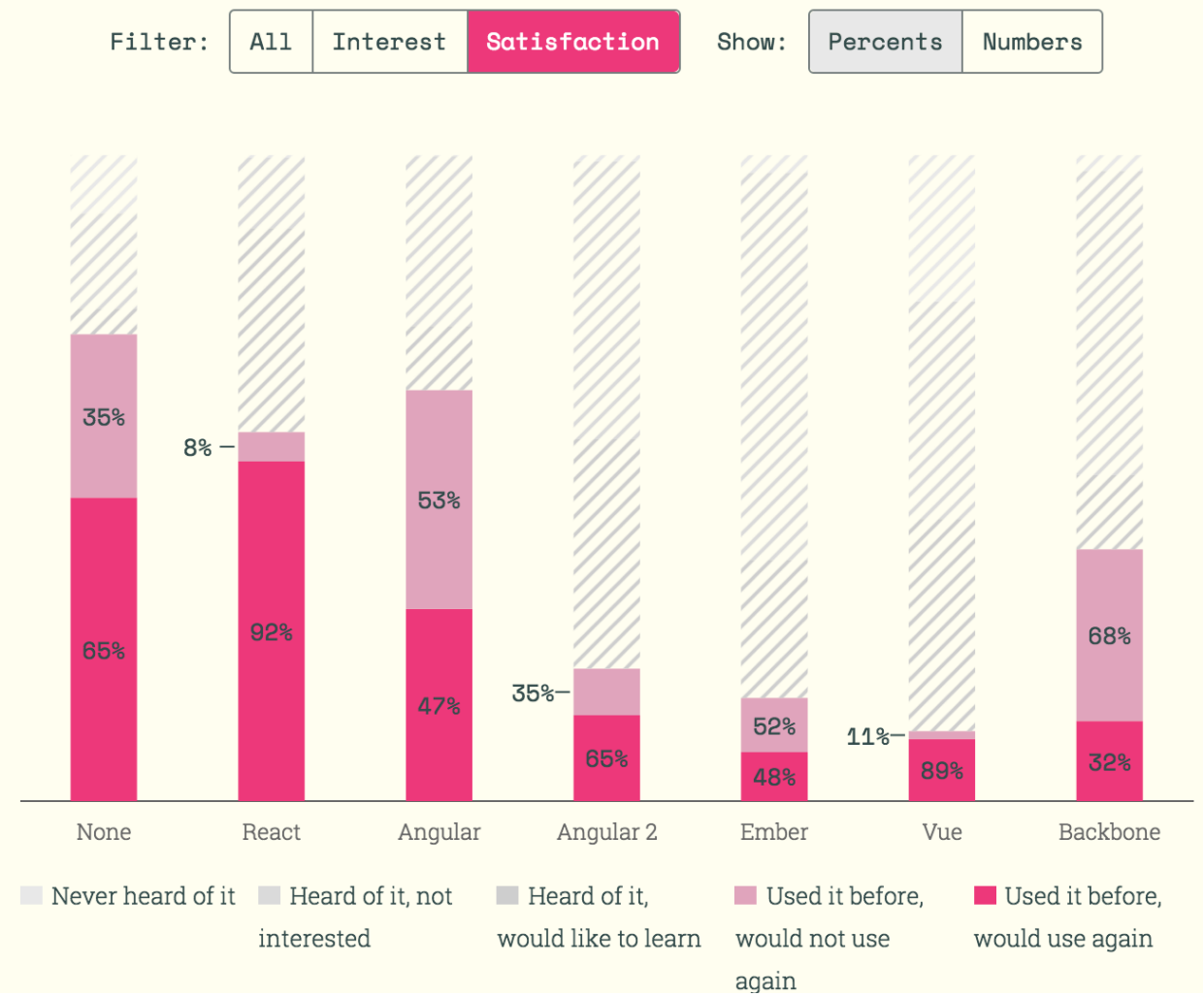
http://stefankrause.net/js-frameworks-benchmark4/webdriver-ts/table.html

Frond-End Solutions Comparison



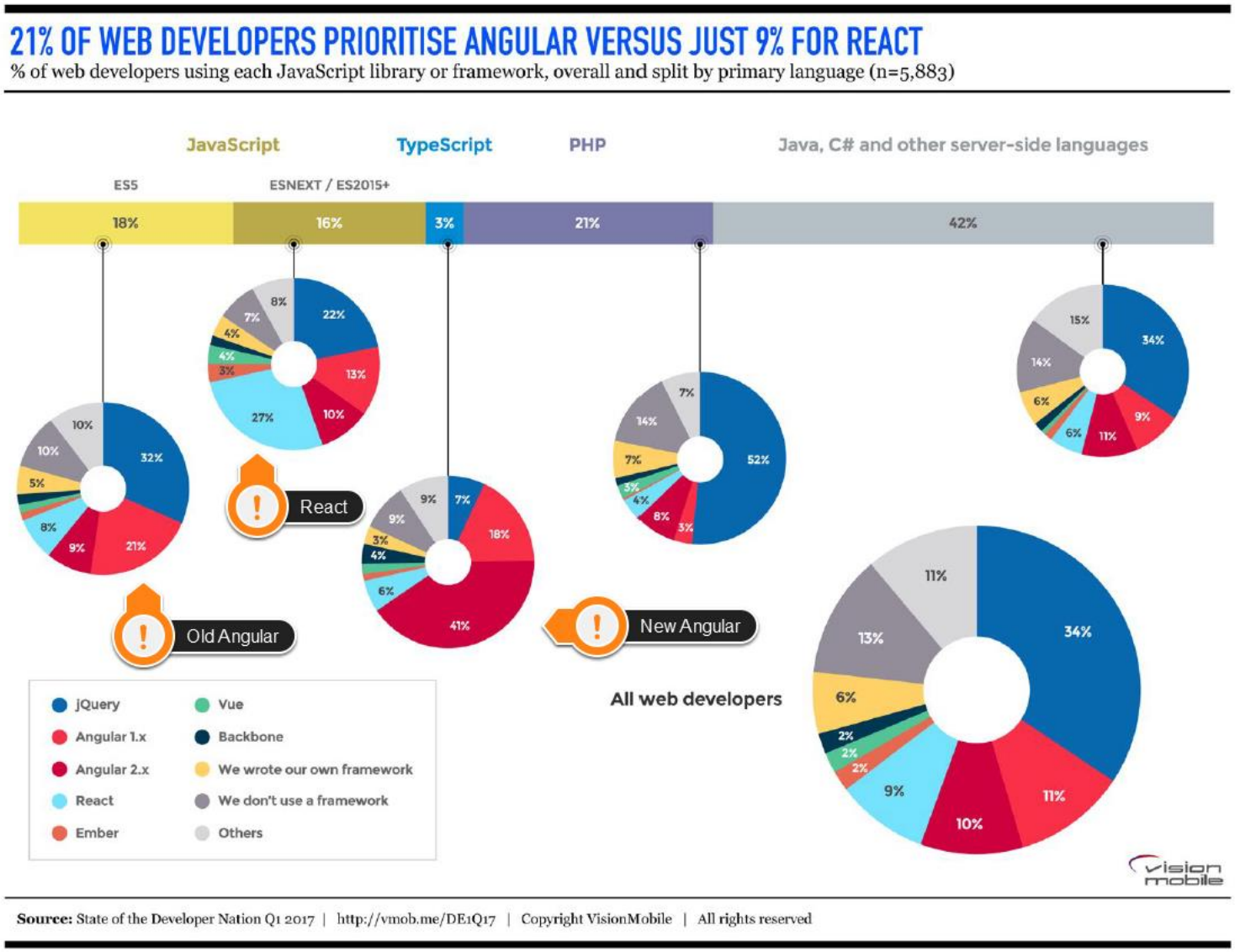
http://stateofjs.com/2016/frontend/

Frond-End Solutions Comparison



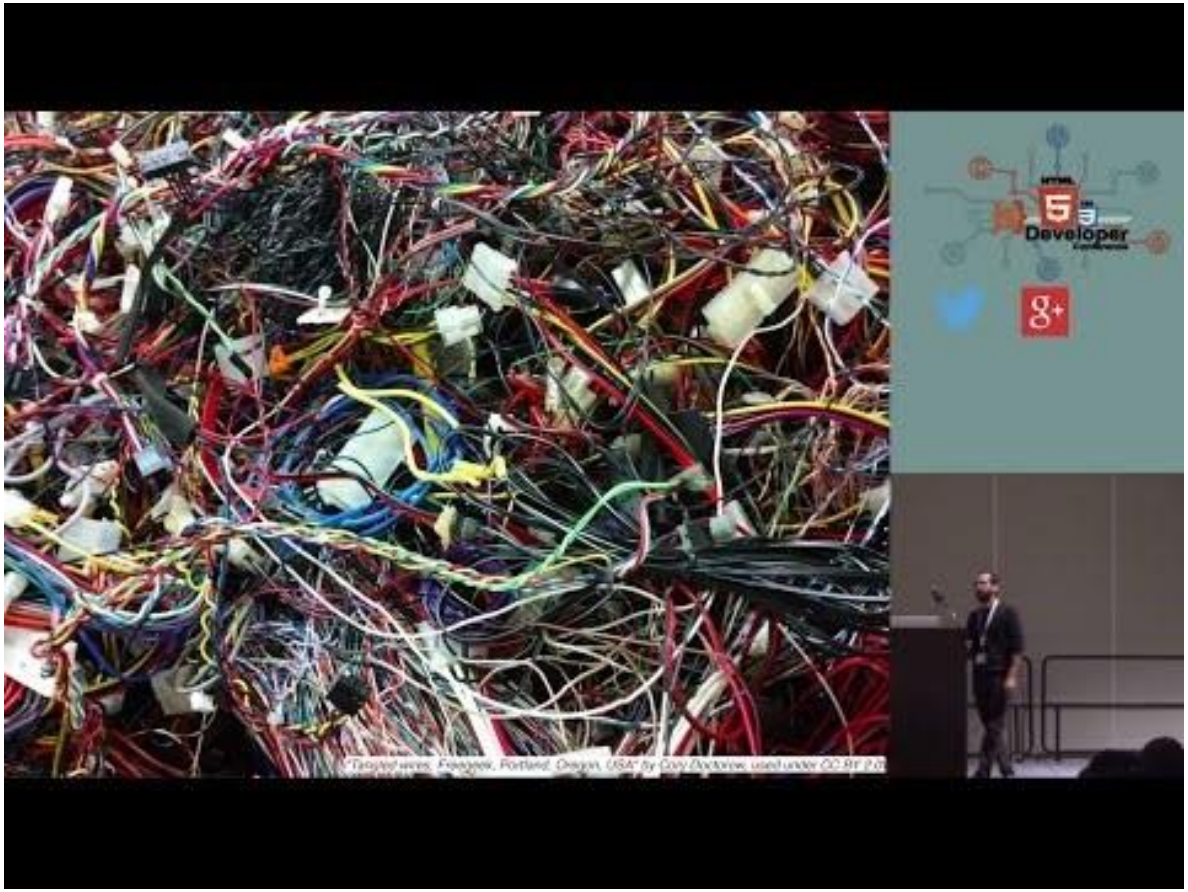
https://wptavern.com/state-of-javascript-survey-results-published-react-emerges-as-clear-winner-in-front-end-frameworks

Frond-End Solutions Comparison



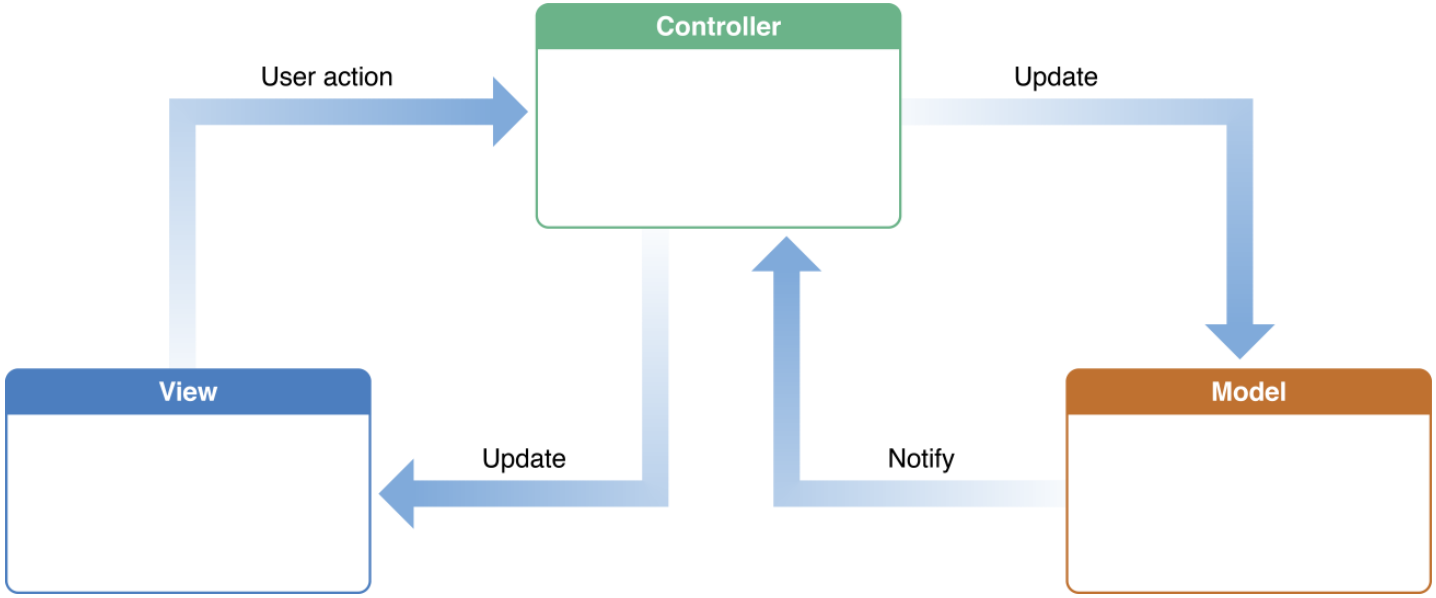
Flux and react.js Concepts

Why current approaches are not sufficient ?



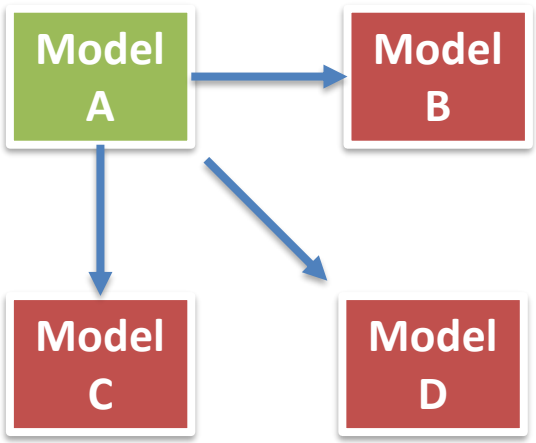
https://www.youtube.com/watch?v=Bic_sFiaNDI

Current State: MVC



https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

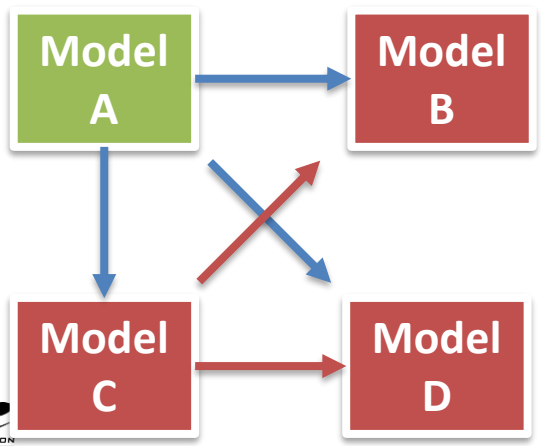
Current State: MVC



- Model B needs (depends on) model A

  → **Model A needs to be updated first and then model B**

- Idem if Model C and Model D needs model A



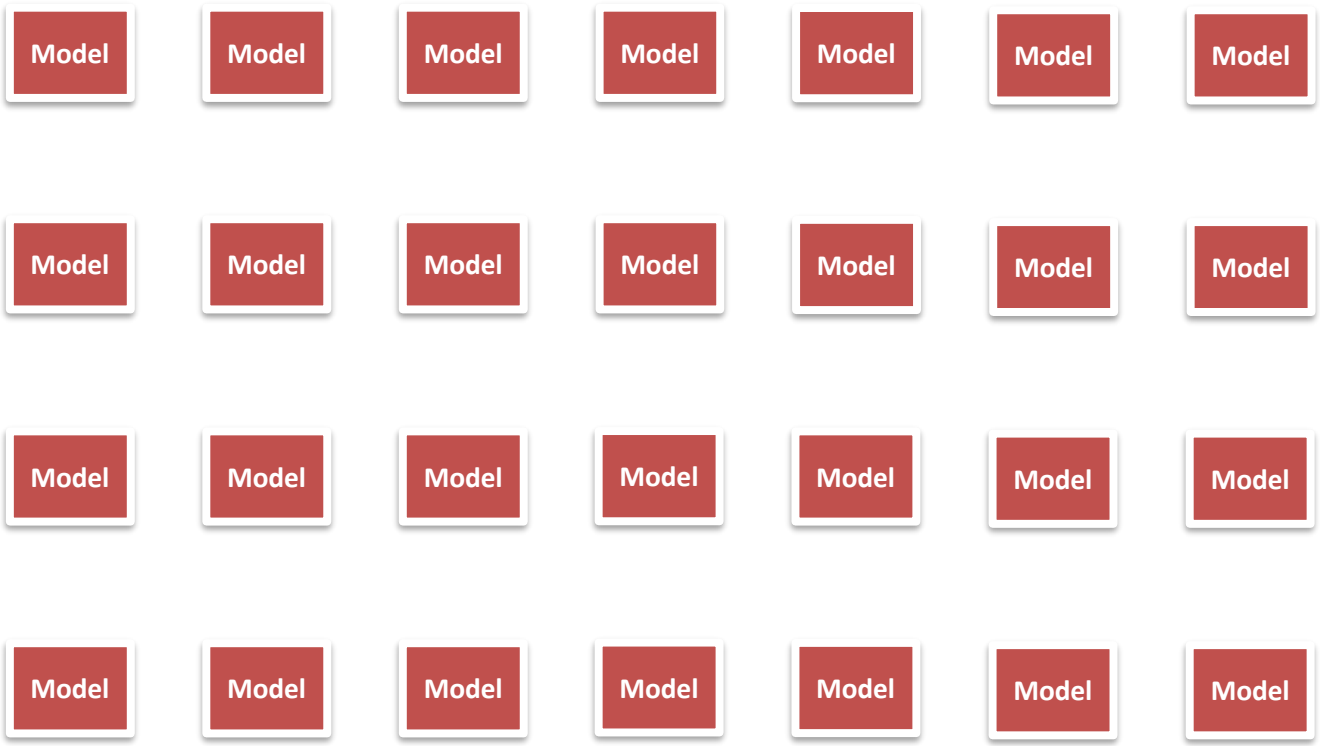
- Model B may needs also on model C

  → **Model A needs to be updated first, then model C and finally Model B**

- **Update needs to be propagated and we need to manage that**

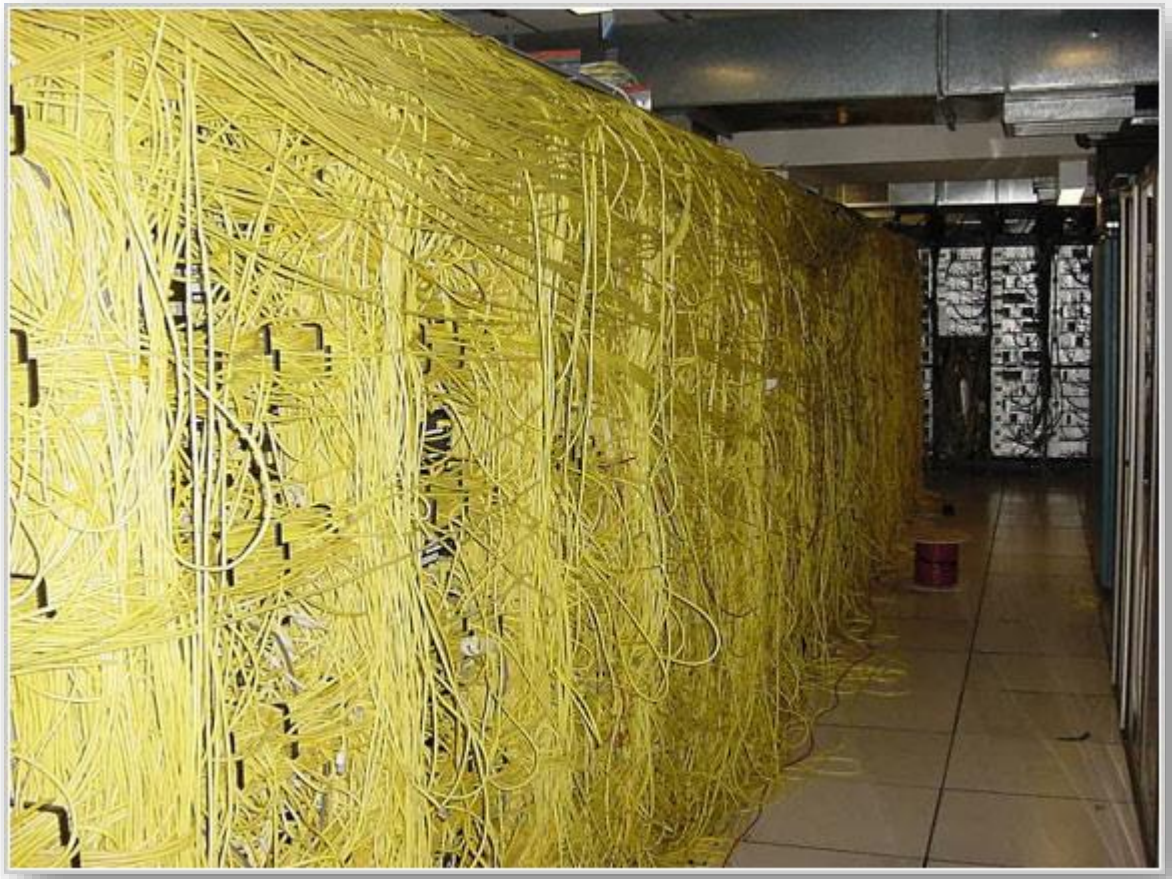
Current State: MVC

What happen when we scale up ?



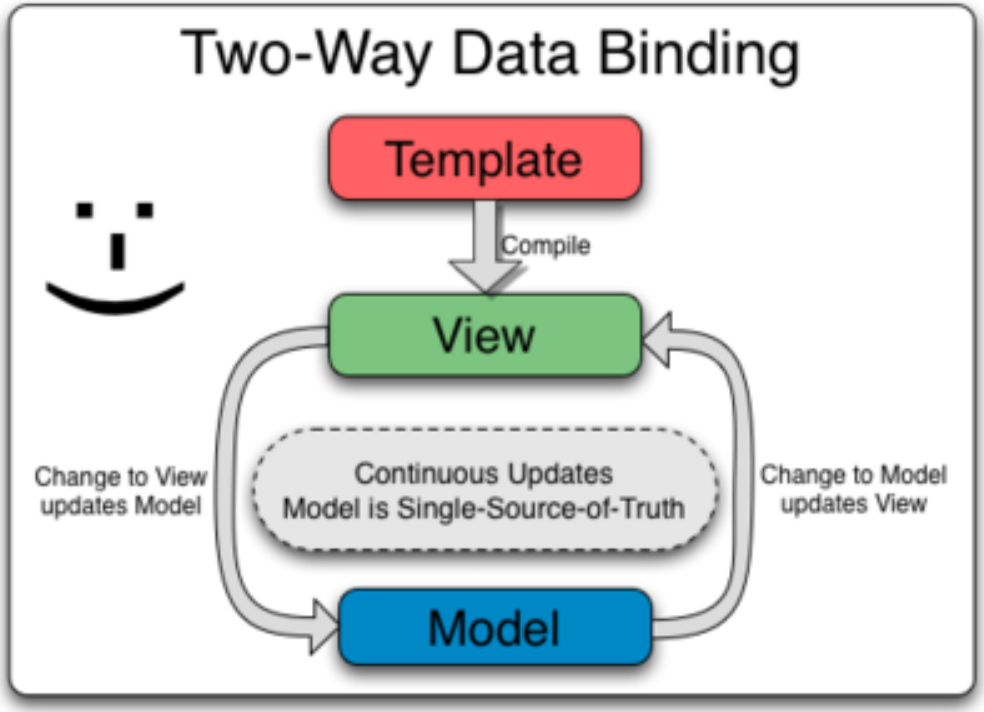
Current State: MVC

What happen when we scale up ?



Current State: MVVM (Model View ViewModel)

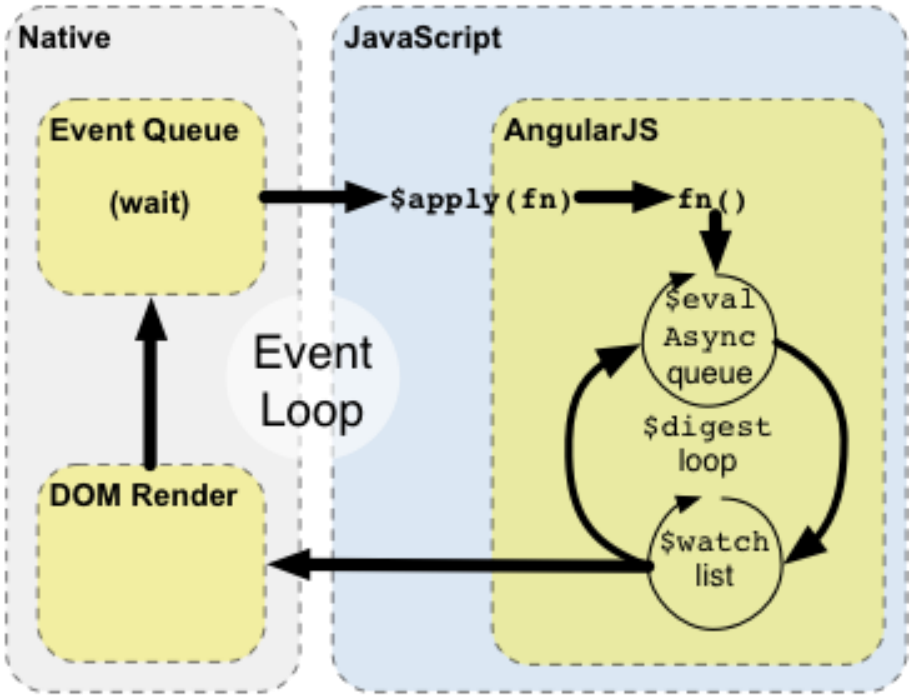
- Angular.js data binding



- **No more controller for managing model update events instead**

Current State: MVVM (Model View ViewModel)

- Angular.js data binding

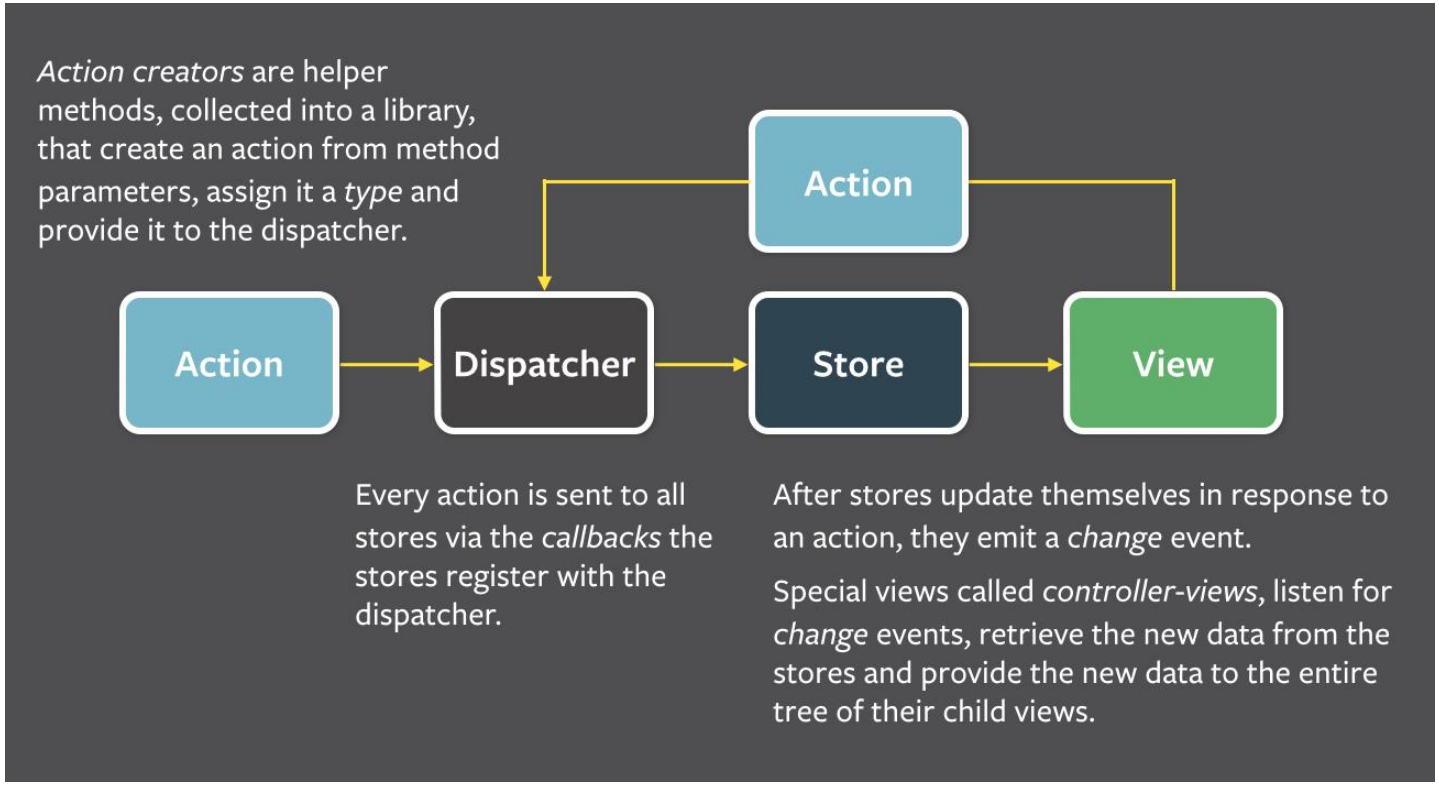


- More details information below

https://docs.angularjs.org/guide/scope#integration-with-the-browser-event-loop

Proposition: Flux

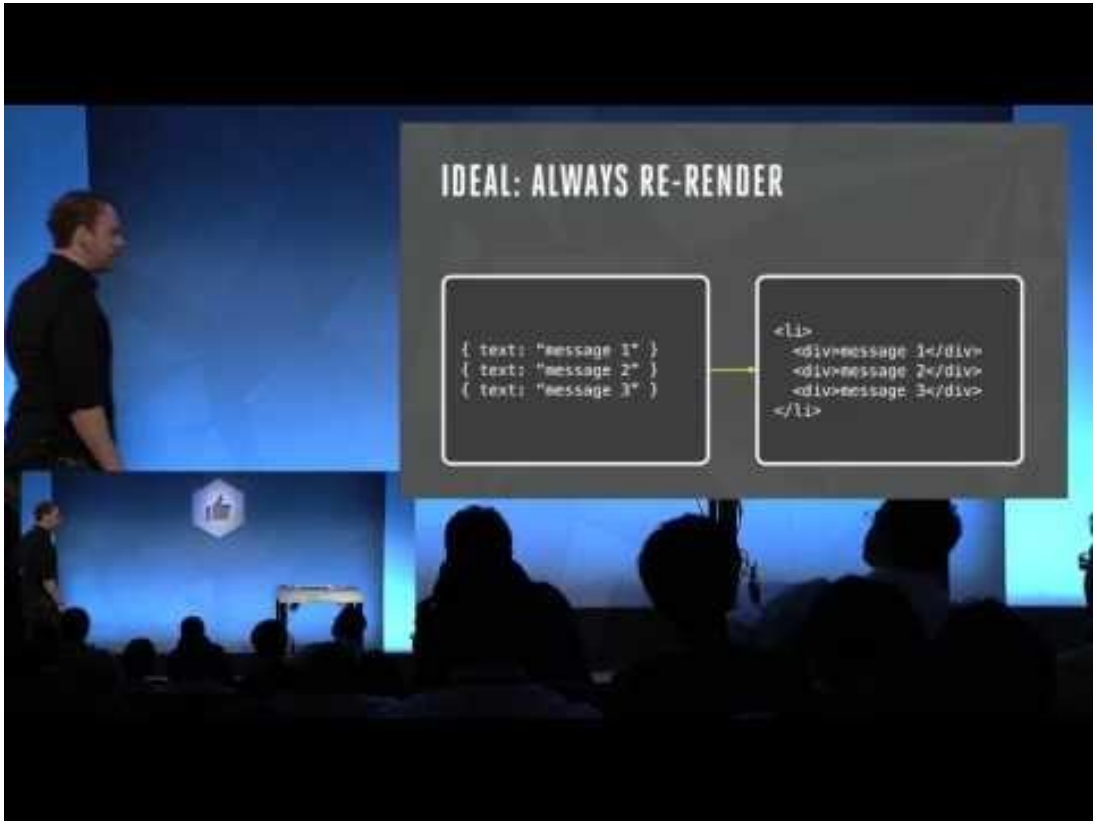
- Single direction data flow: FLUX



https://facebook.github.io/flux/docs/in-depth-overview.html#content

Proposition: Flux

- Details information and explanation here



https://facebook.github.io/flux/docs/in-depth-overview.html#content

UI update Issue

- How updating web page efficiency after data modification ?

```
{ text: 'message 1'}
{ text: 'message 2'}
```

+

```
{ text: 'message 3'}
```

→

```
<li>
    <div>message 1</div>
    <div>message 2</div>
<li>
```

```
Append
<div>message 2</div>
```

UI update Issue

- How updating web page efficiency after data modifications ?

```
{ text: 'message 1'}
{ text: 'message 2'}
{ text: 'message 3'}
```

```
<li>
    <div>message 1</div>
    <div>message 2</div>
    <div>message 3</div>
<li>
```

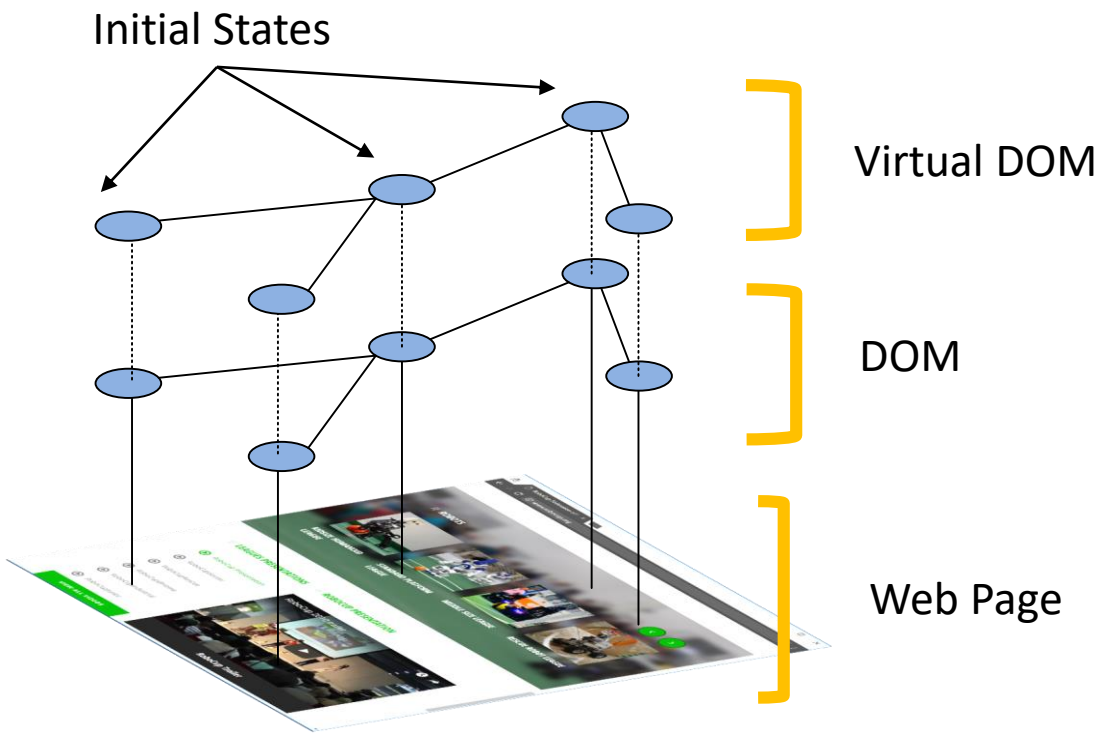
→ We want to always re-render!

Proposition: React.js

- What is React.js?
  - A library for building reusable UI components
  - Implements one-way reactive data flow
  - Mostly use as the V of the MVC.

- React.js Properties
  - Use the Javascript syntax extension (JSX)
  - Optimize the DOM Update through Virutal DOM

React.js : Virtual DOM

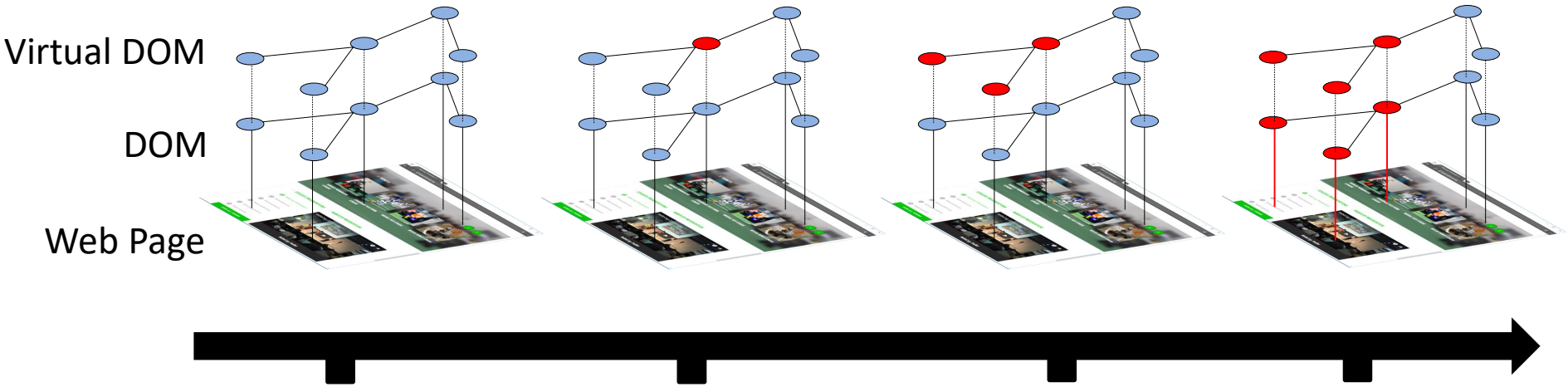


- **Light copy** of the DOM
- **Quick** navigation and update
- Detached from Browser specific

- Hierarchical component view
- Could update and navigate **slowly**
- Browser specific

Ref: http://reactkungfu.com/2015/10/the-difference-between-virtual-dom-and-dom/

React.js : Virtual DOM



Initial State of DOM and Virtual DOM

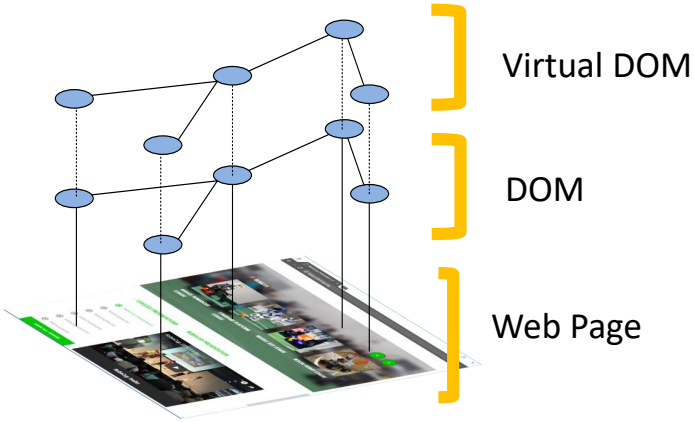
A state changes on the Virtual DOM

A diff is computed between old Virtual Dom and new Virtual DOM

The DOM and web page are re-render according the computed diff

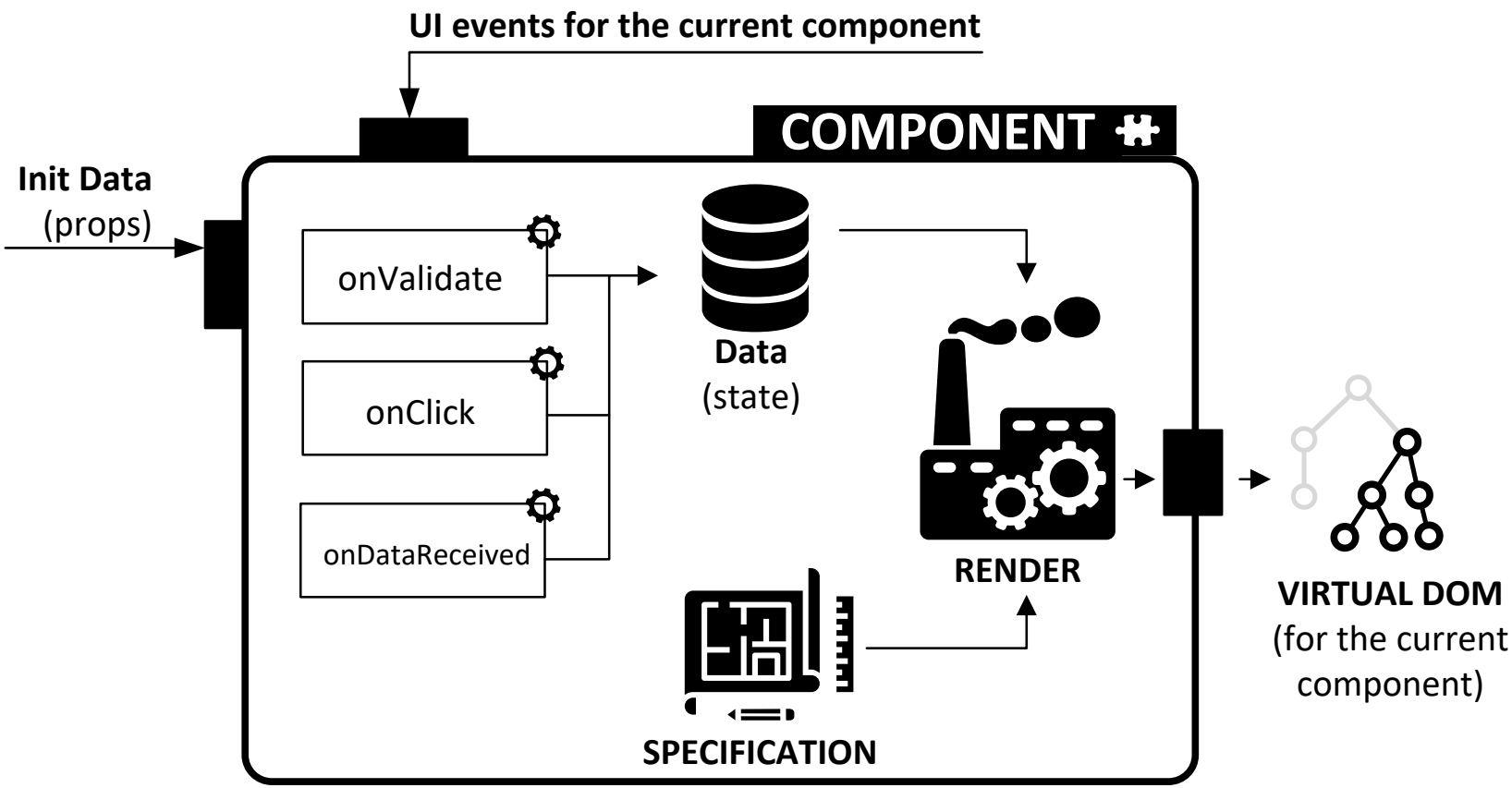
React.js objects

- ReactElement
  - Lowest type of virtual dom
- ReactNode
  - Hierachical Element of the virtual dom
  - ReactElement, string, number
  - Array of virtual nodes
- ReactComponent
  - Specification of how to build react elements



React.js Component

- In react.js everything (mostly) is a component



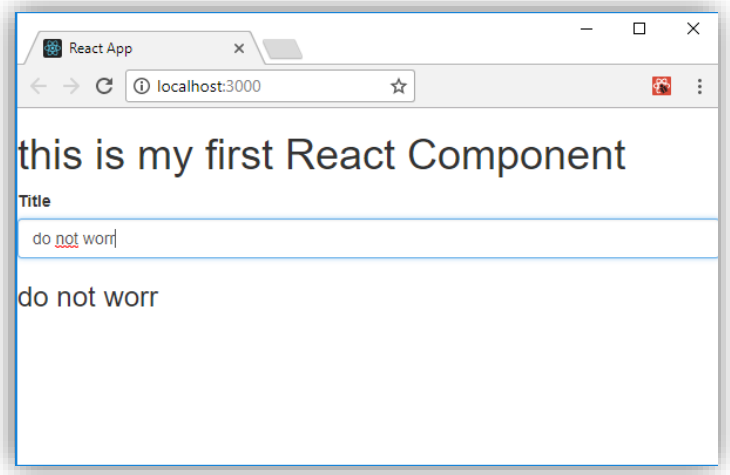
React.js Component

Index.js

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

ReactDOM.render(<App />,
document.getElementById('root'));
```

App.js

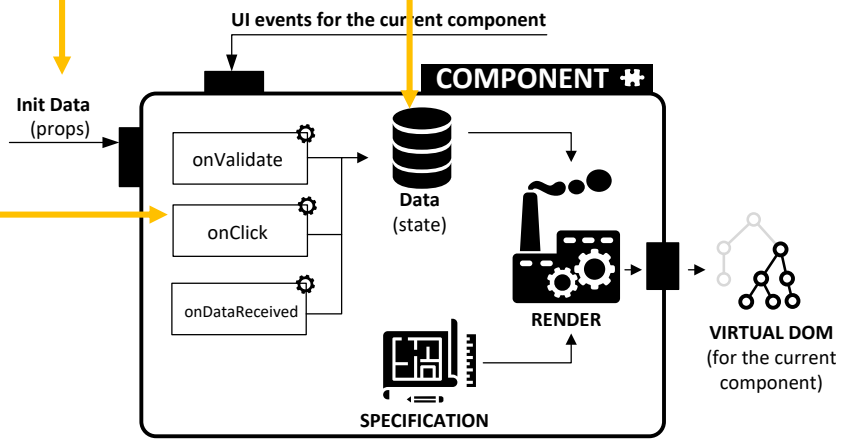
```
import React, { Component } from 'react';
class App extends Component {
  constructor(props) {
  super(props);
  this.state = { title:this.props.title, };
  this.handleChangeTitle=this.handleChangeTitle.bind(this);
   }

 handleChangeTitle(e){
    this.setState({title:e.target.value});
 }

  render() {
   return (
    <div className="App">
     <h1> this is my first React Component</h1>
     <label htmlFor="titleInput">Title </label>
        <input type="text" id="titleInput"
           onChange={this.handleChangeTitle}
           value={this.state.title}
        />
        <h3>{this.state.title}</h3>
    </div>
   ); }}
export default App;
```



React.js Component

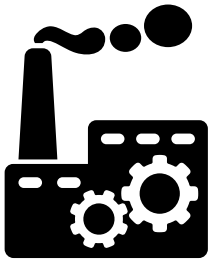
App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor(props) {
  super(props);
  this.state = { title:this.props.title, };
  this.handleChangeTitle=this.handleChangeTitle.bind(this);
   }

 handleChangeTitle(e){
    this.setState({title:e.target.value});
 }

  render() {
   return (
    <div className="App">
     <h1> this is my first React Component</h1>
     <label htmlFor="titleInput">Title </label>
        <input type="text" id="titleInput"
           onChange={this.handleChangeTitle}
           value={this.state.title}
        />
        <h3>{this.state.title}</h3>
    </div>
   );  }}
export default App;
```



React.js From JSX to JS


```
var profile = <div>
  <img src="avatar.png" className="profile" />
  <h3>{[user.firstName, user.lastName].join(' ')}</h3>
</div>;
```

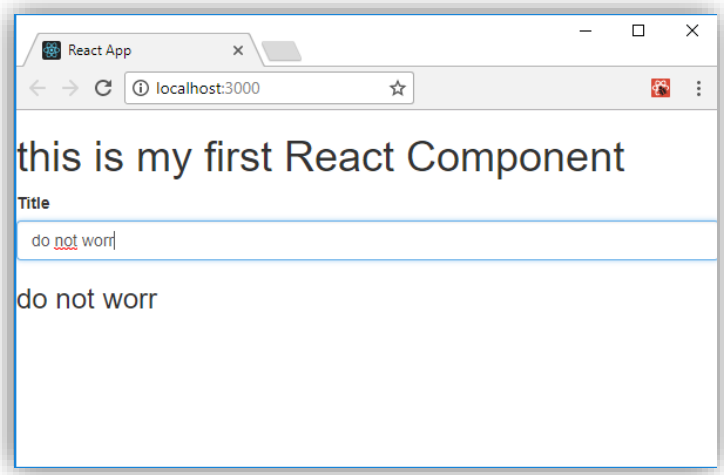


RENDER


```
var profile = React.createElement("div", null,
  React.createElement("img", { src: "avatar.png", className: "profile" }),
  React.createElement("h3", null, [user.firstName, user.lastName].join(" "))
);
```

More information herehttps://jasonformat.com/wtf-is-jsx/

React.js Component



Application DOM

```
<html lang="en">
  <head>
      <meta charset="utf-8">
      <title>React App</title>
  </head>
  <body style="">
     <div id="root">
        <div data-reactroot="" class="App">
            <h1> this is my first React Component</h1>
            <label for="titleInput">Title </label>
            <input type="text" class="form-control" id="titleInput">
               <h3></h3>
        </div>
     </div>
  </body>
</html>
```

Syntax

- Javascript based on the **ES6 standard**
- Object syntactic sugar
- New symbols usage (generator, arrow, functions)
- Set of libraries (promises, new collections, types arrays)

Syntax ES6

From http://slidedeck.io/DonaldWhyte/isomorphic-react-workshop

CLASSES

```
class Point {
  constructor(x, y) {
      this.x = x; this.y = y;
   }
   toString() { return `(${this.x}, ${this.y})`; }
}
```

IMPORTING OTHER MODULES

Modules are JavaScript files.

```
// import foo.js in same dir as current file import foo
from './foo';
foo.foobar(42);

// import specific variables/functions from a module
import { foobar } from './foo'; foobar(42);
```

LET and CONST

```
const x_const
Let x_var

 x_var = x_var +1
x_const = x_const +50 // → error
```

INHERITENCE

```
class ColorPoint extends Point {
    constructor(x, y, color) {
        super(x, y);
        this.color = color;
    }
    toString() { return super.toString() + ' in ' + this.color; }
}
```

EXPORTING SYMBOLS

foo.js:

```
export function foobar(num) { console.log('FOOBAR:', num); }
```

EXPORTING ENTIRE OBJECTS

person.js:

```
export default { name: 'Donald', age: 24 };
```

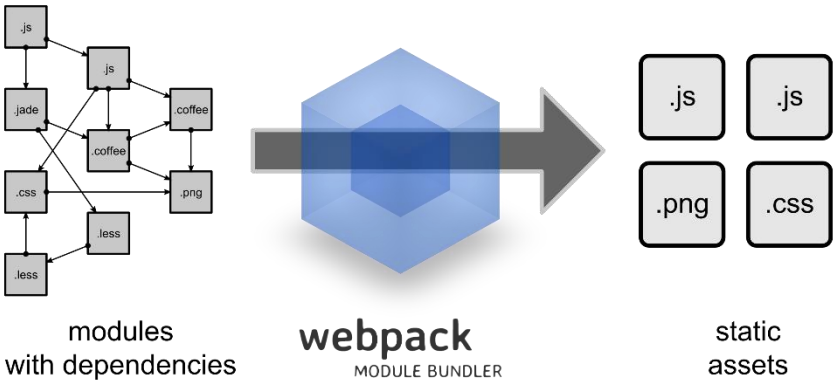
another_file.js

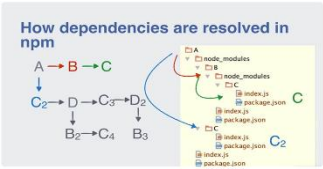
```
import person as './person'; console.log(person.name);
// outputs 'Donald'
```

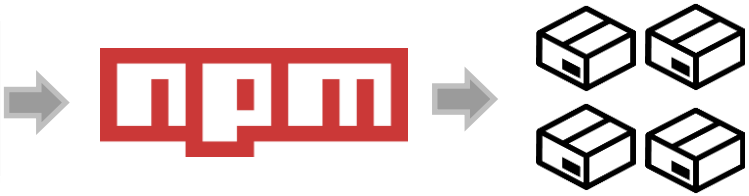
FUNCTION

```
()=>{return 'hello';}
```

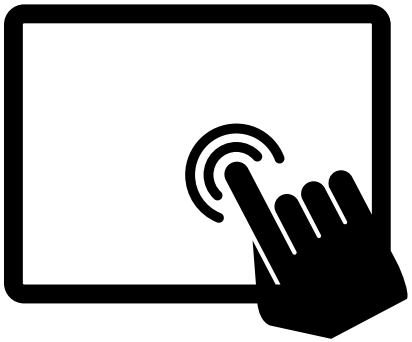
React ToolBoxe : Configure a read.js project







https://www.codementor.io/tamizhvendan/beginner-guide-setup-reactjs-environment-npm-babel-6-webpack-du107r9zr



React.js : Let's go!

Installation

https://facebook.github.io/react/docs/installation.html

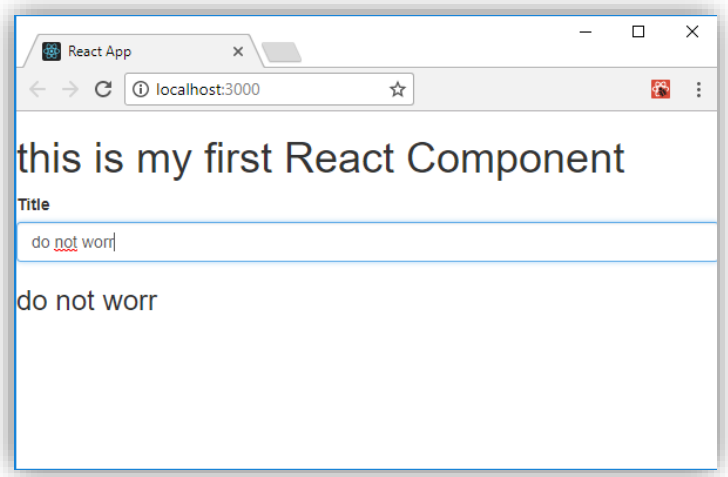
- Nodes.js
- Npm

```
npm install -g create-react-app
create-react-app my-app
cd my-app
npm start
```

```
npm run build
```

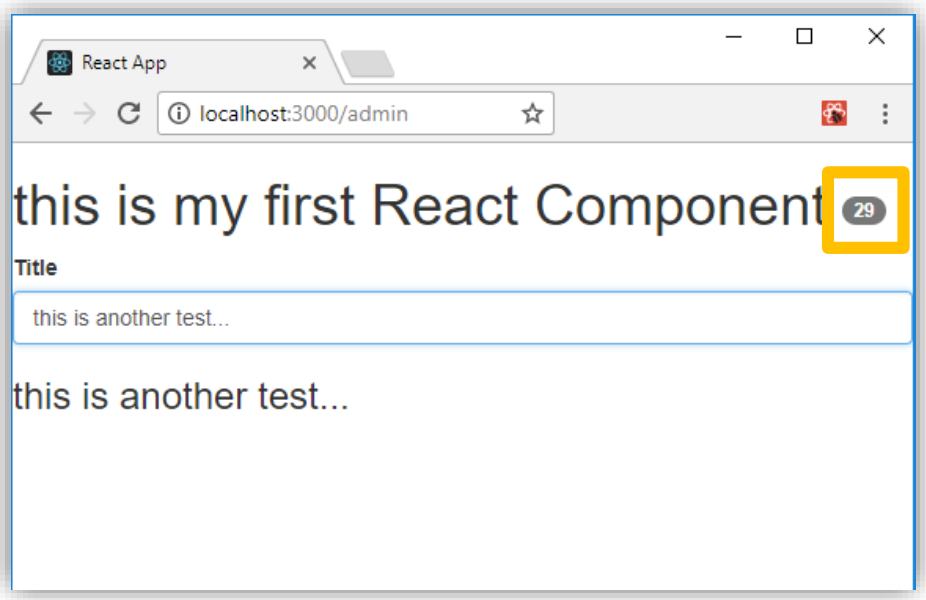
It is your turn !

- Create App allowing to get as input a title and print it below
- Your App component must be initialized with the title property = 'default_title '



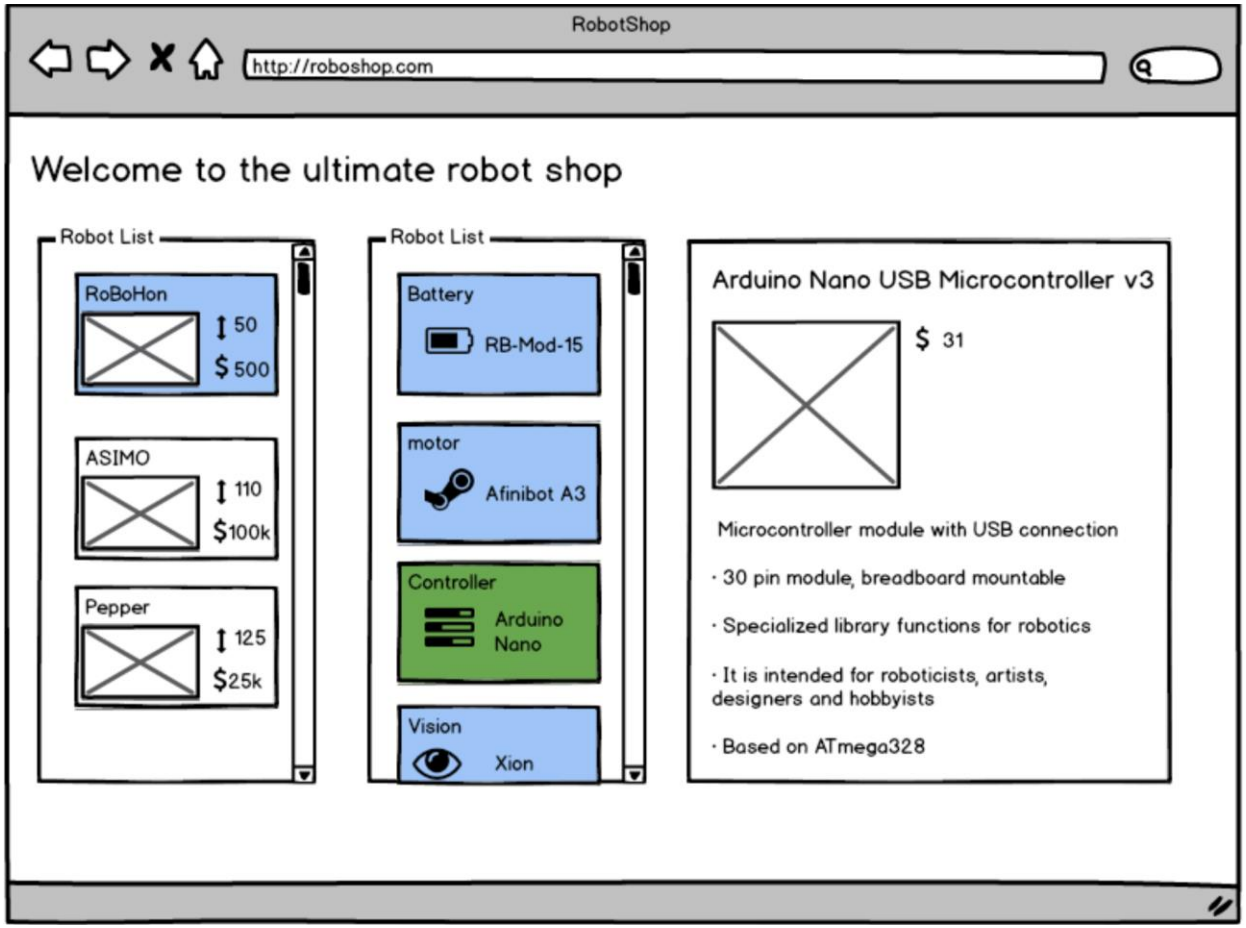
It is your turn !

- Update the number of mouse over the printed title



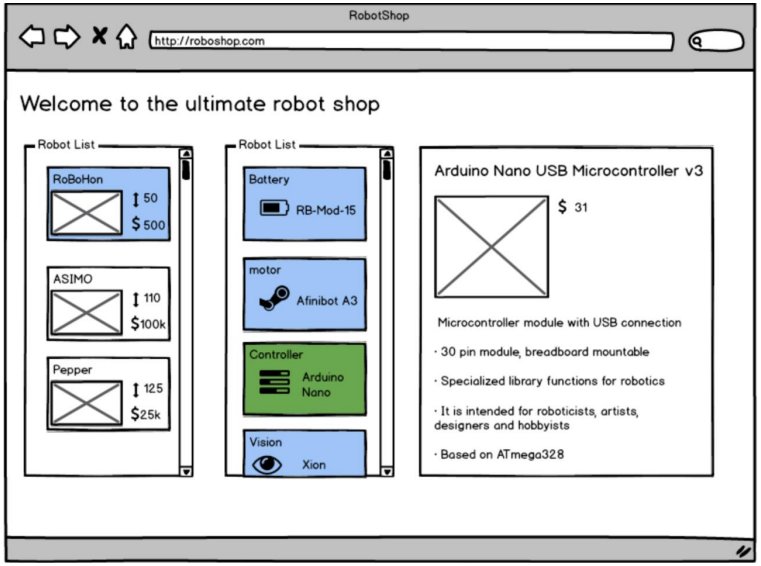
Best practices

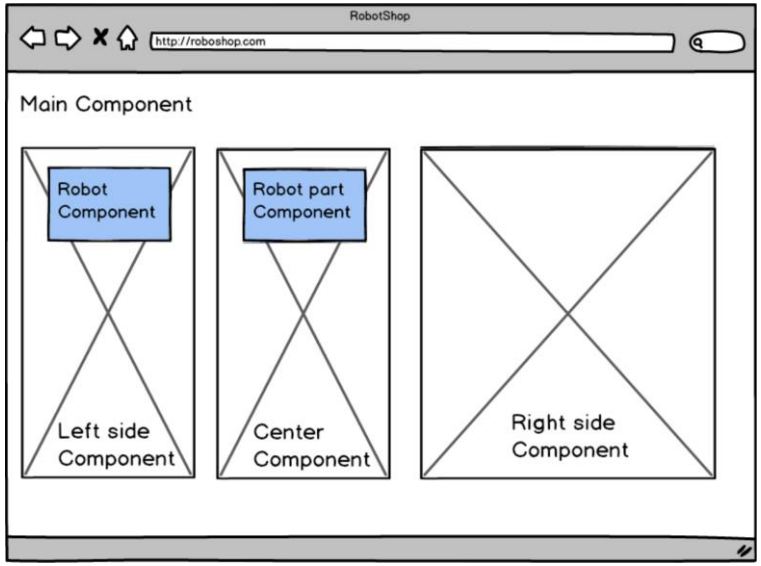
- Divide your application into components



Best practices

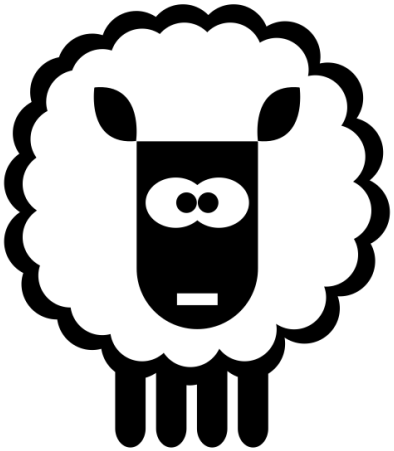
- Divide your application into components



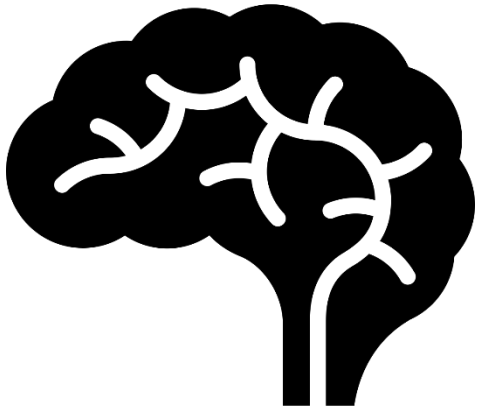


Best practices

- **Displaying** (presentational Component) **Vs Processing** (container Component)

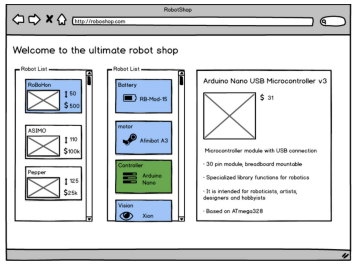


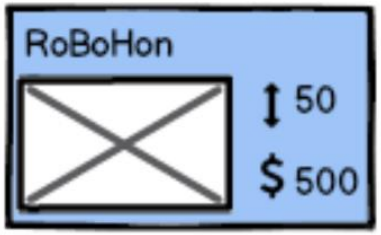
VS

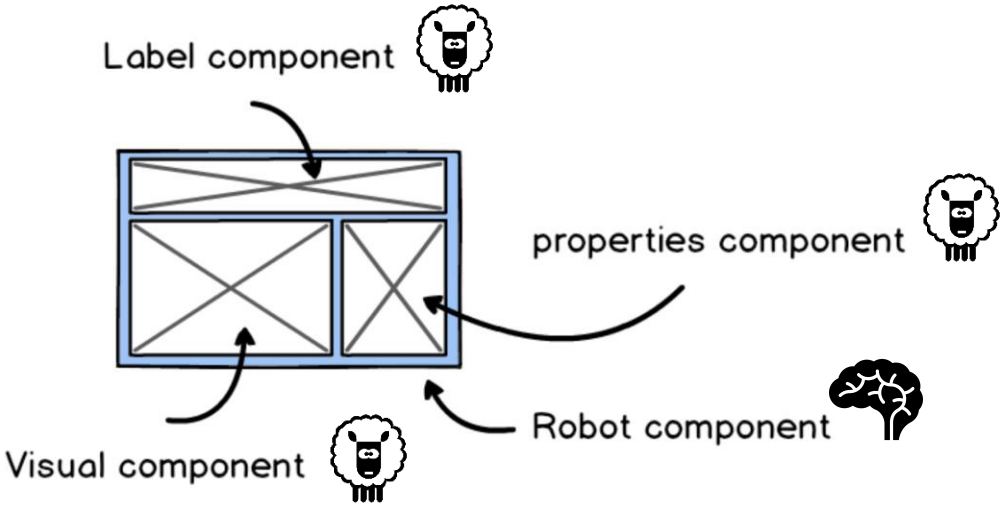


Best practices

- Displaying Vs Processing

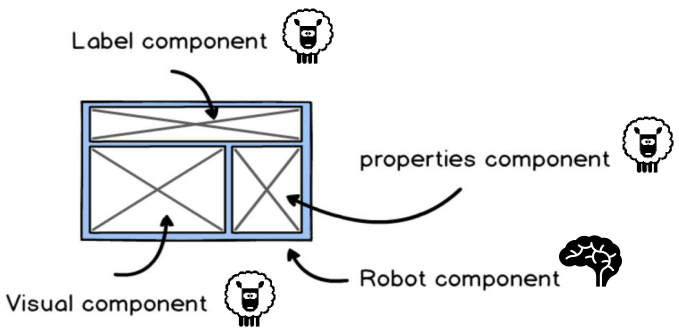




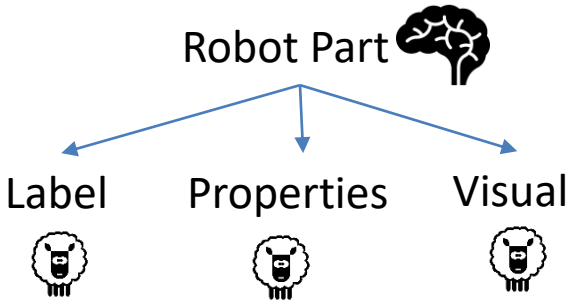


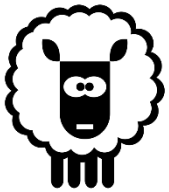
Best practices

- Displaying Vs Processing



Process Data

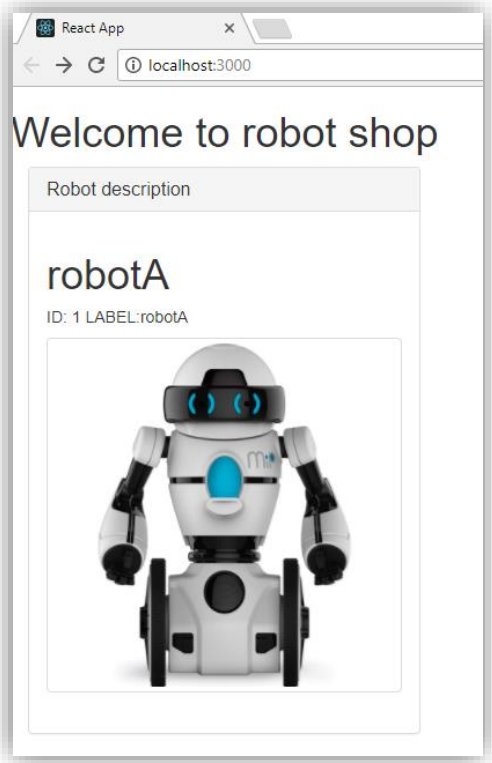




Display Data

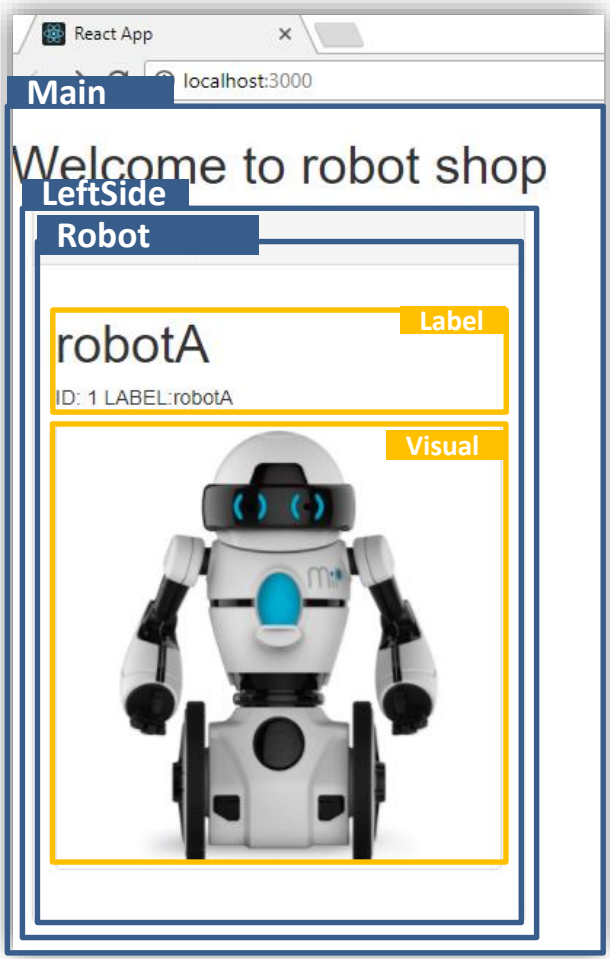
It is your turn !

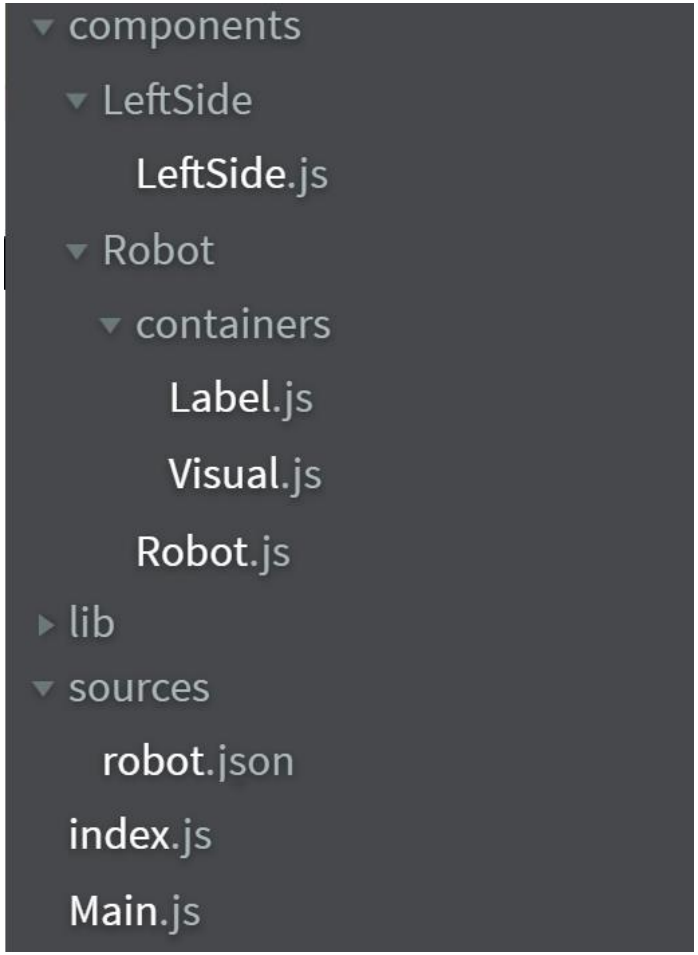
- Create the following application
- Create
  - A main component initialized with a json file
  - A Left side component
  - A Robot component
  - A Label component for the Robot Component



All steps available at:
https://github.com/jacques-saraydaryan/front-end-react.js.git

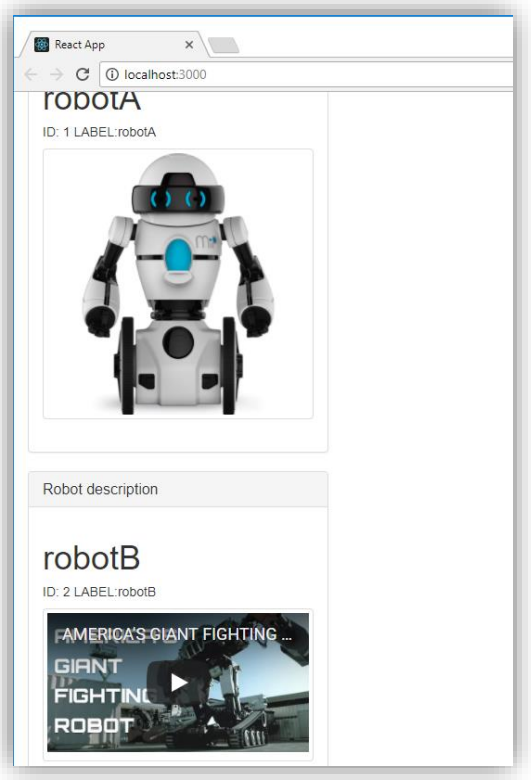
It is your turn !





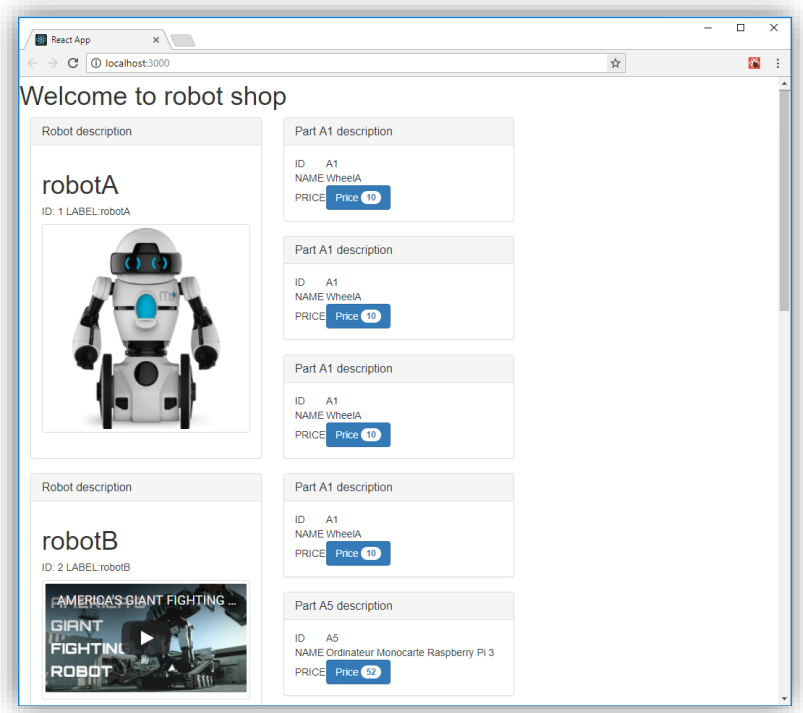
It is your turn !

- Same as previously but with a list of robots
- Some robot can have a visual as a video

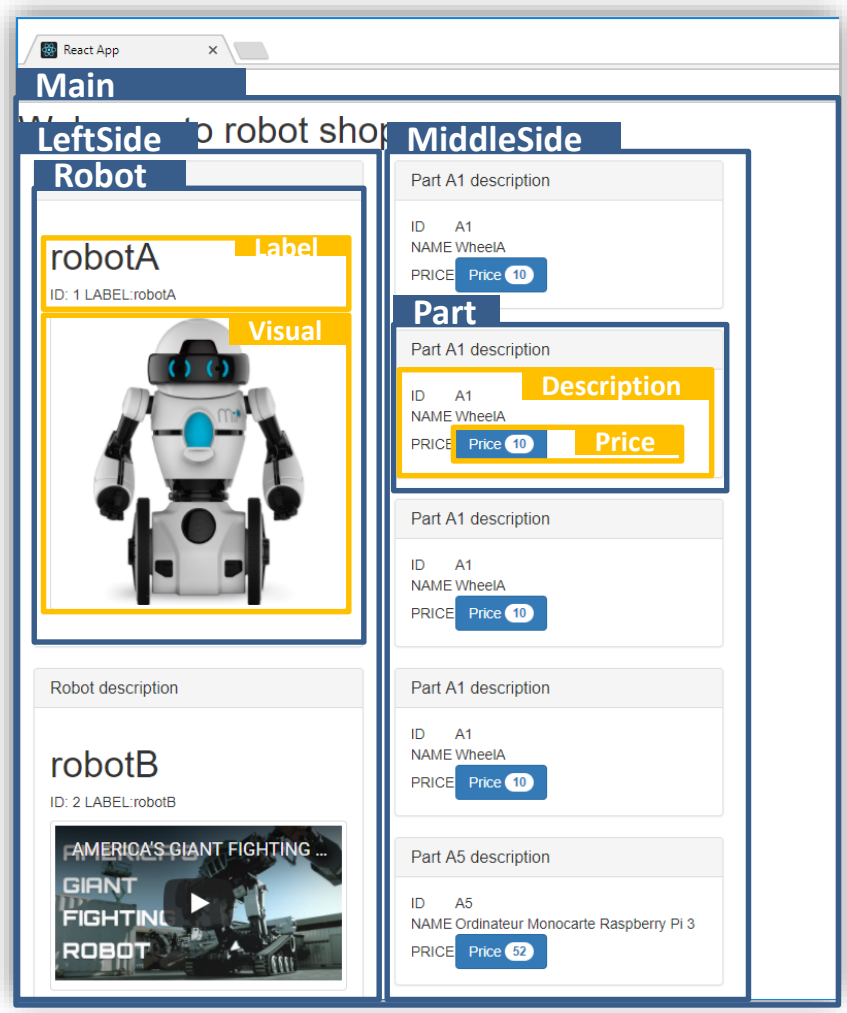


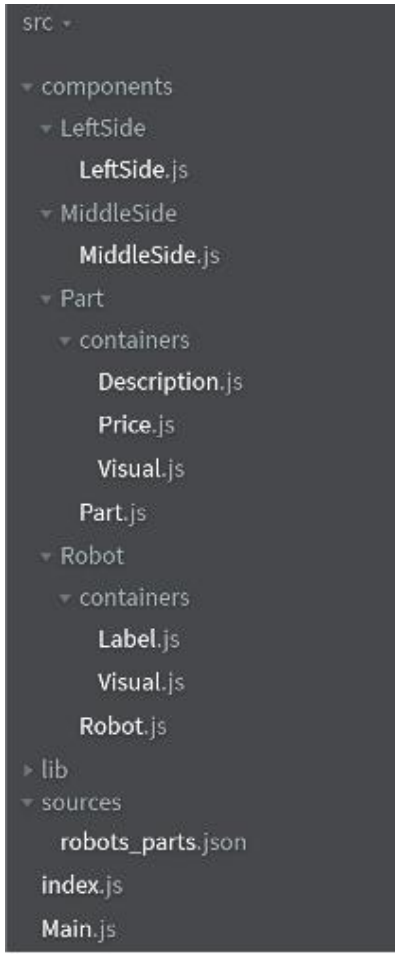
It is your turn !

- Add a Part Component using
  - Description Component
  - Price Component
  - Visual Component
- Add a MiddleSide Component displaying the list of parts of the selected robot



It is your turn !





It is your turn !

Robots_parts.json

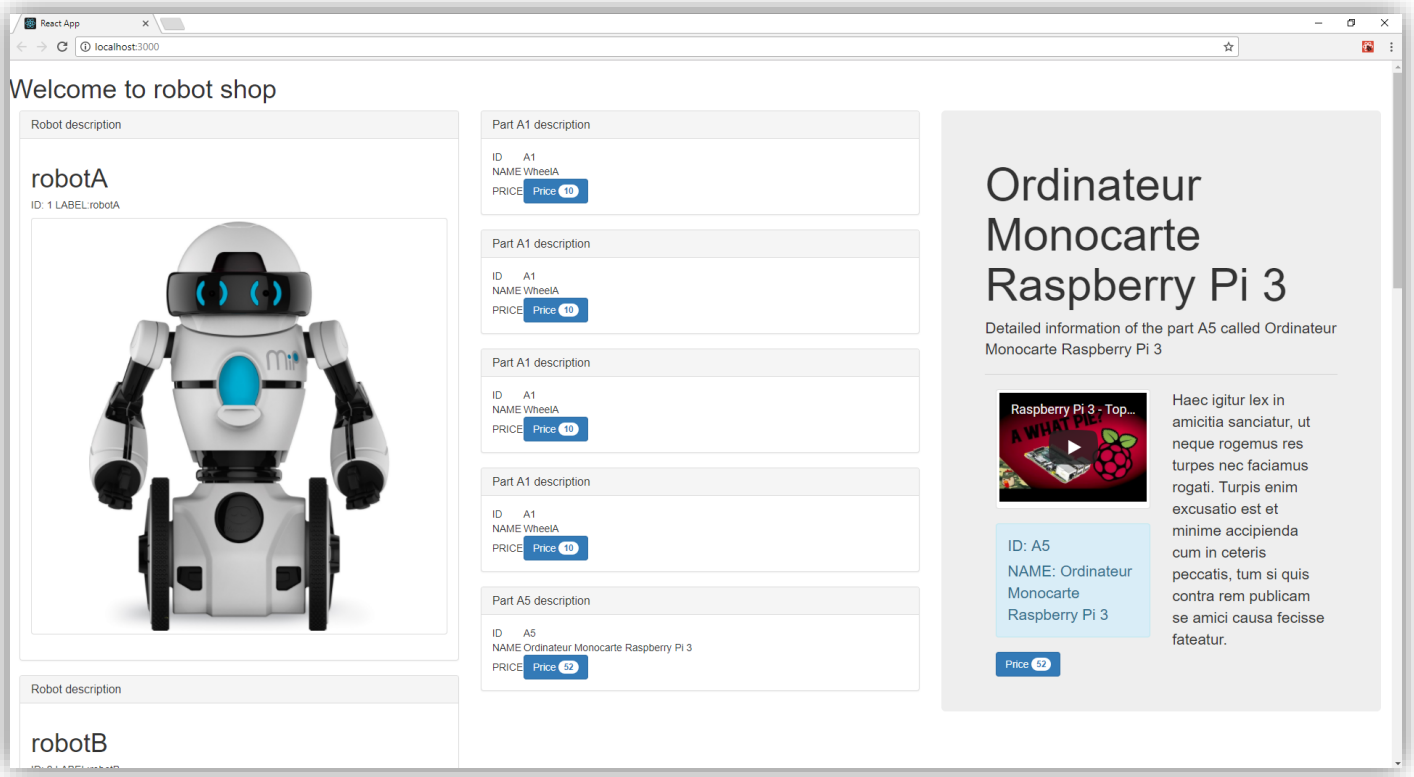
```
{"robots":[
  {
    "id":1,
    "title":"robotA",
    "visual_type":"img",
    "visual_src":"https://www.robot-advance.com/EN/ori-wowwee-
mip-white-robot-1281_1613.jpg",
    "parts":["A1","A1","A1","A1","A4"]
  },
  {
    "id":2,
    "title":"robotB",
    "visual_type":"video",
    "visual_src":"https://www.youtube.com/embed/ePINYZK4p5Y",
    "parts":["A2","A2","A4"]
  },
],
```

```
"parts":[
  {
    "id":"A1",
    "title":"WheelA",
    "price":10
  },
  {
    "id":"A2",
    "title":"WheelB",
    "price":15
  },
  {
    "id":"A3",
    "title":"WheelC",
    "price":150
  },
  {
    "id":"A4",
    "title":"Contrôleur de Servomoteurs USB SSC-32U Lynxmotion",
    "price":57
  } ]
}
```

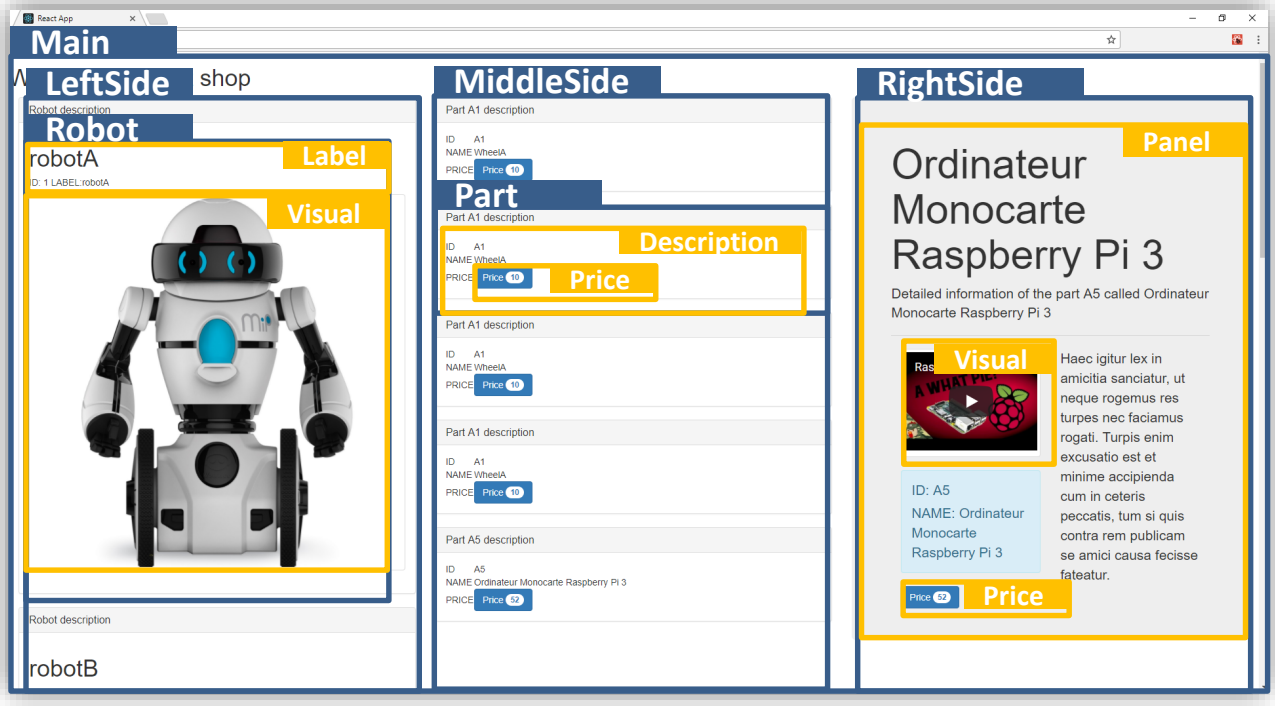
```
Npm install json-loader
```

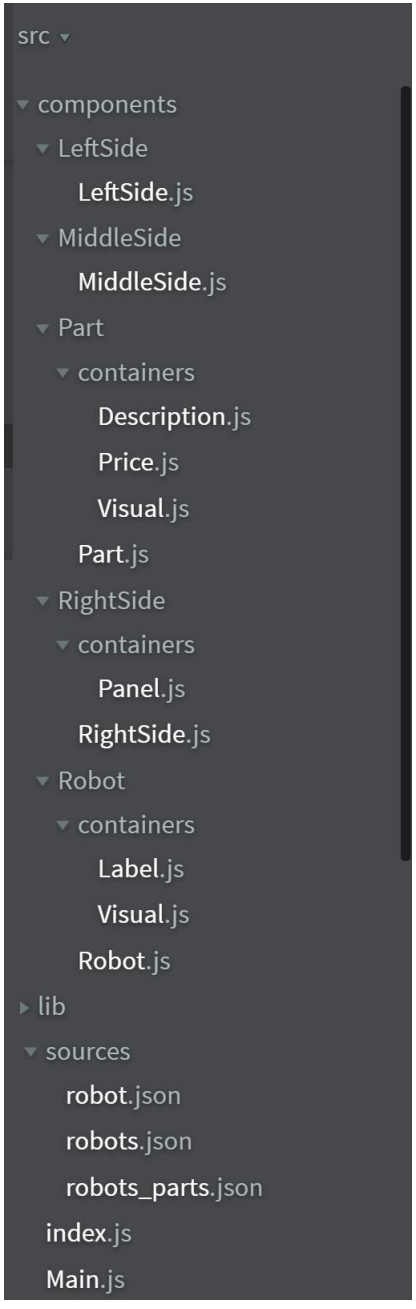
It is your turn !

- Add a RightSide Component using
  - Panel Component displaying selected part



It is your turn !

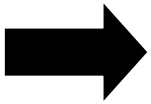




React.js :
Our current State
Redux as enhancement

React.js and so?

- **Pro**
  - Components Based
  - Extremely efficient (Virtual Dom)
  - Easy to write module base code
  - UI Test Cases easy to write
- **Con**
  - Only for the view layer
  - Write visual component into Javascript
  - Hard to learn
  - **Hierarchical dependencies !**



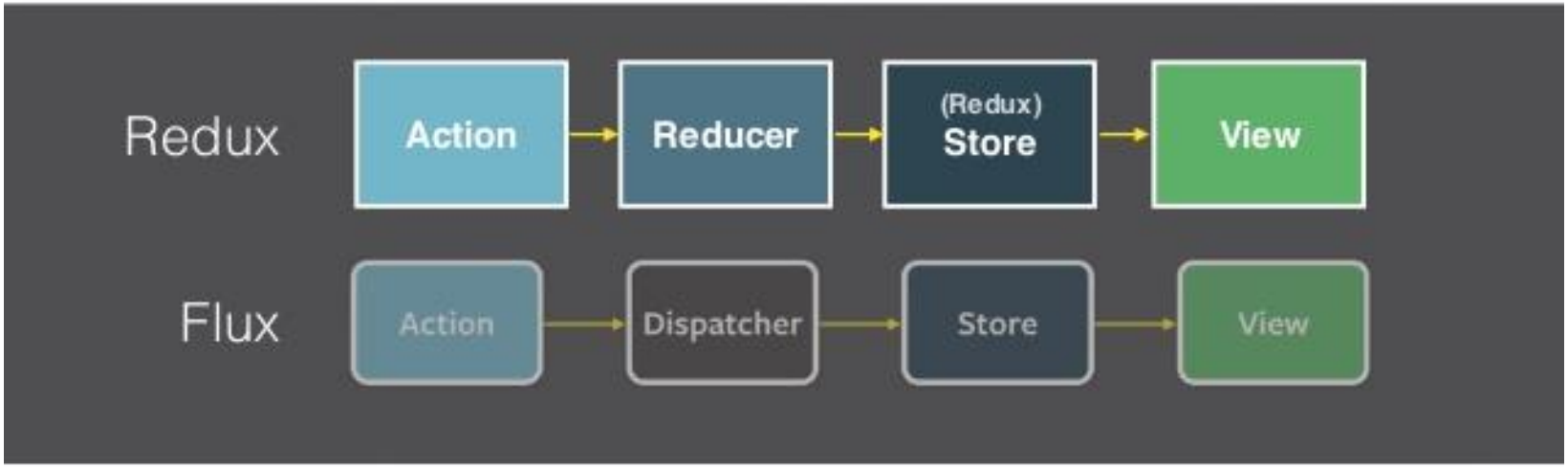
Redux

Flux

Redux

Can be view as an implementation of Flux

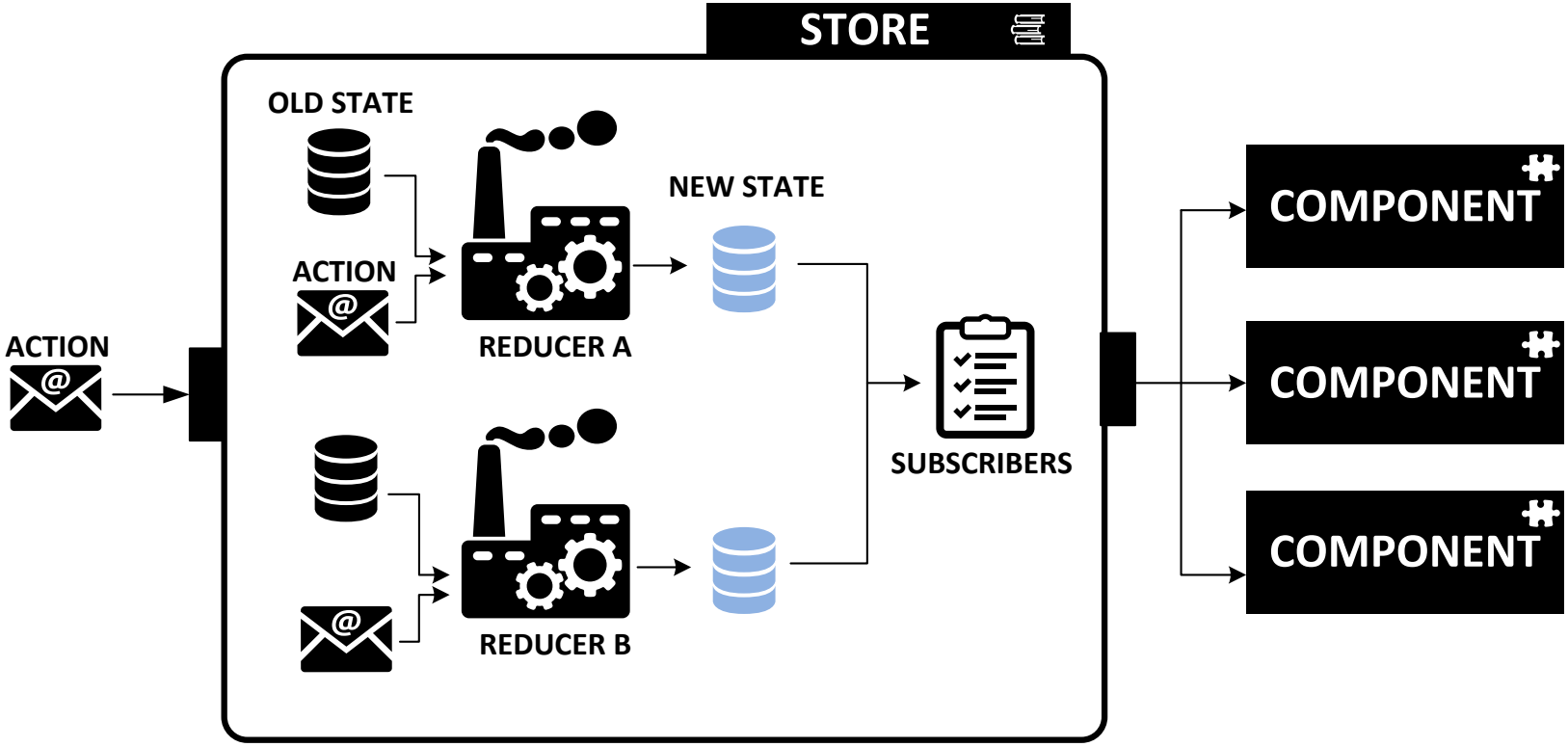
Redux



Redux vs traditional Flux

https://www.slideshare.net/JonasOhlsson/using-redux

Redux

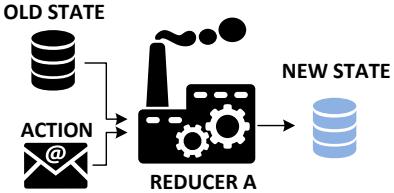

STORE
OLD STATE
ACTION
REDUCER A
NEW STATE
ACTION
REDUCER B
SUBSCRIBERS
COMPONENT
COMPONENT
COMPONENT

Redux

1. Define actions

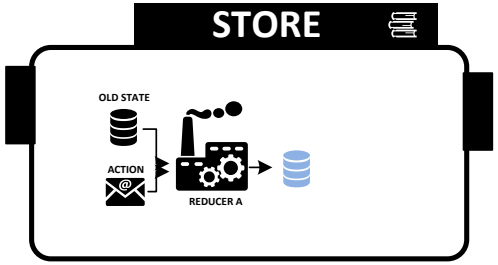

```
export const myAction =
        () => {
         return { type: 'A', value: 2 };
        }
```

2. Define a Reducer



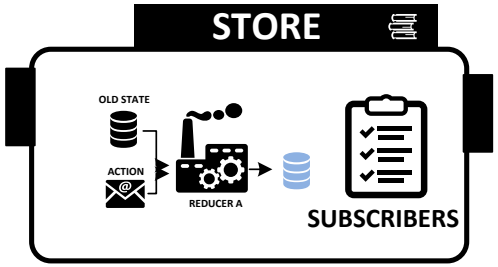

```
function app(state = 0, action) {
  switch (action.type) {
    case 'A': return state + action.value
    case 'B': return 0
    default: return state
  }
}
```

3. Create A Store with the Reducer




```
import { createStore } from 'redux';

const store = createStore(app);
```

4. Component register to the Store




```
store.subscribe(() => console.log(store.getState()) )
```

Redux in practice (1/4)

- Install reduce components

```
npm install redux

npm install react-redux
```

- Create actions and reducers into dedicated files and directories

```
src
  actions
    index.js
  reducers
    index.js
    partReducer.js
    robotReducer.js
  components
  lib
```

- Use Global reducers merging several reducers

index.js

```
import { combineReducers } from 'redux';
import robotReducer from './robotReducer';
import partReducer from './partReducer';

const globalReducer = combineReducers({
  robotReducer: robotReducer,
  partReducer: partReducer
});

export default globalReducer;
```

robotReducer.js

```
const robotReducer= (state={},action) => {
    console.log(action);
    switch (action.type) {
        case 'UPDATE_SELECTED_ROBOT':
            return action.obj;
    default:
      return state;
    }
}

export default robotReducer;
```

Redux in practice (1/4)

Troubleshooting

http://redux.js.org/docs/Troubleshooting.html

This is a place to share common problems and solutions to them.

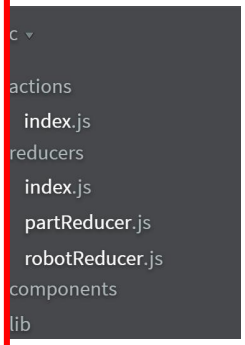
The examples use React, but you should still find them useful if you use something else.

Nothing happens when I dispatch an action

Sometimes, you are trying to dispatch an action, but your view does not update. Why does this happen? There may be several reasons for this.

Never mutate reducer arguments

It is tempting to modify the `state` or `action` passed to you by Redux. Don't do this!




```
import { combineReducers } from 'redux';
import robotReducer from './robotReducer';
import partReducer from './partReducer';

const globalReducer = combineReducers({
  robotReducer: robotReducer,
  partReducer: partReducer
});

export default globalReducer;
```

```
const robotReducer= (state={},action) => {
    console.log(action);
    switch (action.type) {
        case 'UPDATE_SELECTED_ROBOT':
            return action.obj;
    default:
      return state;
    }
}

export default robotReducer;
```

Redux in practice (2/4)

- Use react-redux tools
  - ***Provider*** to deliver service to all components

```
import { Provider } from 'react-redux';
…
```

```
 render() {
    return (
    <Provider store={store} >
     <div className="container-fluid">
       <div className="row">
           <h1> Welcome to robot shop</h1>
       </div>
       <div className="row">
           <div className="col-md-4 col-lg-4" >
               <LeftSide
                   robots={this.state.robot_list}
               />
                       …
           </div>
           </div>
     </div>
    </Provider>
    );
  }
```

Redux in practice (3/4)

- Use react-redux tools
  - *Connect* to dispatch action, and subscribe to modifications

dispatch action

```
//load the connect tool
import { connect } from 'react-redux';
//load the custom action
import {setSelectedRobot} from '../../actions';

class Robot extends Component {
...
handleOnRobotSelected(robot_obj){
           //get the store contained into props and
           // 'sent' an action
    this.props.dispatch(setSelectedRobot(robot_obj));
}
 ...
//link the Robot Component to the store provided by the Provider Tool
//No need to subscribe that's why there is no first parameter to the
// connect function connect()(Robot)
export default connect()(Robot);
```

Redux in practice (4/4)

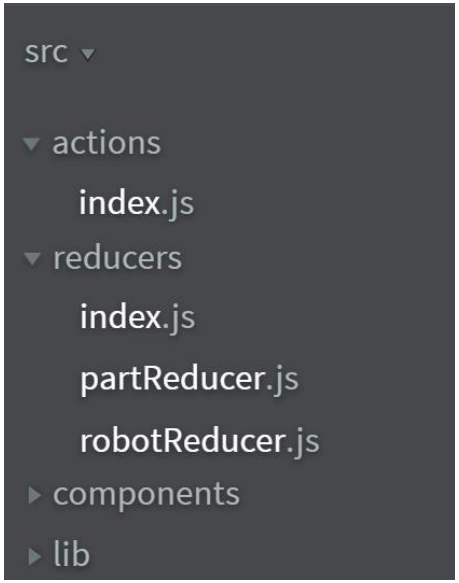
- Use react-redux tools
  - *Connect* to dispatch action, and subscribe to modifications

subscribe to modifications

```
//load the connect tool
import { connect } from 'react-redux';
class MiddleSide extends Component {
            ...
  //render function use to update the virtual dom
  render() {
    return (
          <div>
            <Part
                  part={this.props.parts[0]}
                />
          </div>
   ); } }
const mapStateToProps = (state, ownProps) => {
  return {
    parts: state.robotReducer.parts
  } };

//connect the MiddleSide component to the store provided by the Provider Tool
// Specify a function (mapStateToProps) allowing to trig on store state modification
// when a store state is modify the mapStateToProps is launched and associate the
state.robotReducer.parts value to the local propery parts (this.props.parts)
export default connect(mapStateToProps)(MiddleSide);
```

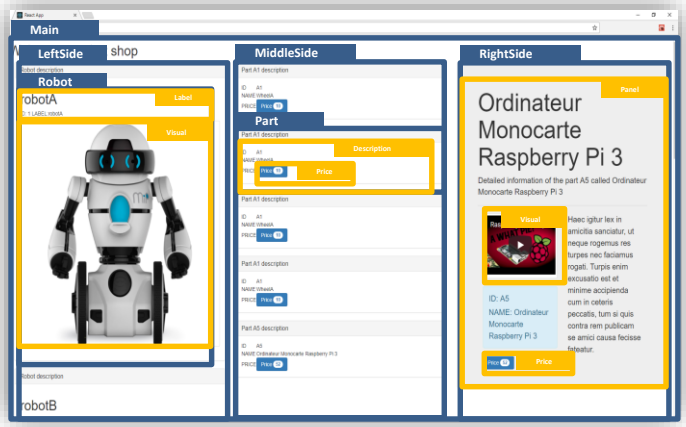
It is your turn !

- Create 2 Actions modifying the selected Robot obj and the selected Part obj
- Create 1 reducer ‘robotReducer’ in charge of processing the selected Robot obj
- Create 1 reduce ‘partReducer’ in charge of processing the selected Part obj



It is your turn !

- Modifying the previous project so as to
  - Dispatch selected robot in the ***Robot*** component
  - Subscribe to store and update list of parts in the ***MiddleSide*** component
  - Dispatch selected part in the ***Part*** component
  - Subscribe to store and update Part in the ***RightSide*** component



References

References

- React.js and Flux
  http://soat.developpez.com/tutoriels/javascript/architecture-flux-avec-react/
  https://www.tutorialspoint.com/reactjs/reactjs_using_flux.htm
  https://medium.com/@jetupper/hello-react-js-b87c63526e3a
  DEVOX France: https://www.youtube.com/watch?v=IFM8krjbKmQ

- React.js and Redux
  http://www.troispointzero.fr/2016/03/reactjs-redux-pour-des-lendemains-qui-chantent-premiere-partie/
  https://www.codementor.io/mz026/getting-started-with-react-redux-an-intro-8r6kurcxf
  http://www.sohamkamani.com/blog/2017/03/31/react-redux-connect-explained/

- Tutorials
  https://github.com/HurricaneJames/dex/blob/master/doc/Building%20Components%20with%20React.js%20and%20Flux.md
  https://github.com/react-bootcamp/react-workshop

- Course tutorial
  https://github.com/jacques-saraydaryan/front-end-react.js

Jacques Saraydaryan

Jacques.saraydaryan@cpe.fr