SECTION 2:
WHAT IS REACT.JS?

WHAT IS REACT.JS?

- Simply put, it is a JavaScript library for building User Interfaces (UI) created and maintained by Facebook.
- ReactJS makes no assumptions about your technology stack, so you can use ReactJS to:
  - Build a widget
  - Add a reusable component (header, footer, etc.)
  - Build the entire front-end experience (like Facebook)
- Just to reiterate, you can incorporate ReactJS into different types of front-end tech stacks (AngularJS, Backbone, etc.) or you can choose to build entire applications out of ReactJS!

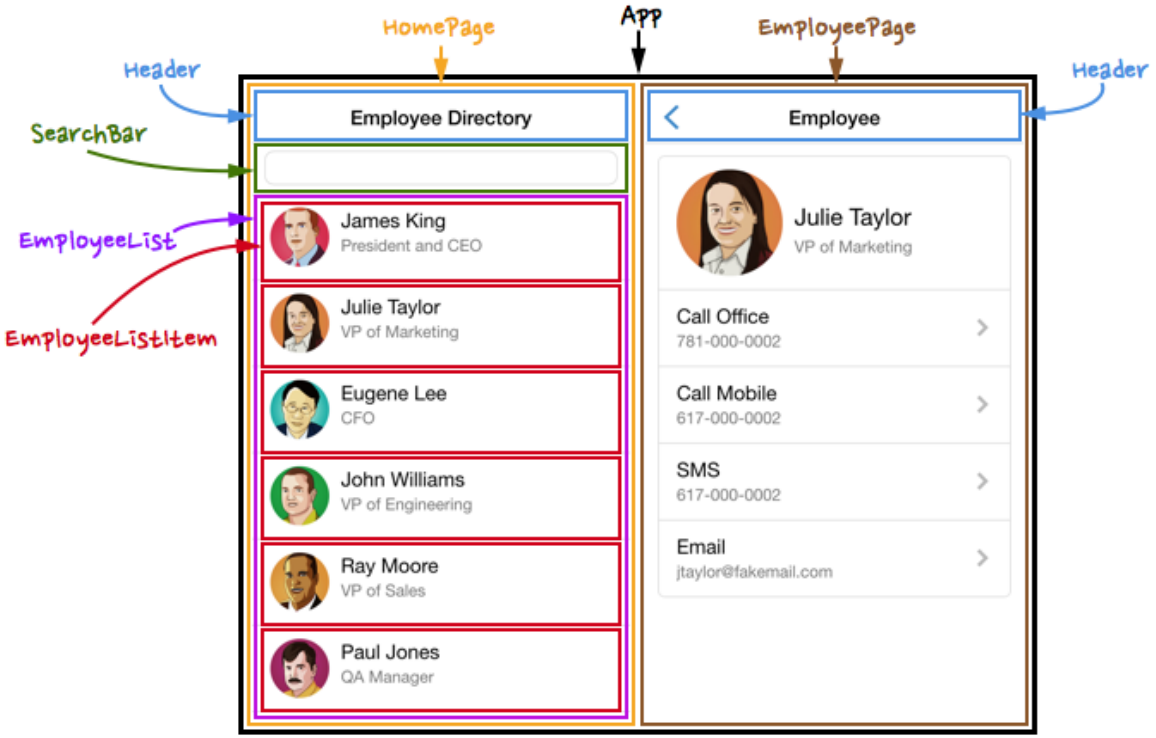
WHAT IS REACT.JS?

- React.js community is extremely huge. At this moment React has ~40000 Stars on Github
- Around it people built whole universe with Redux (the most popular flux implementation), GraphQL, Relay, tools like hot reloading and time machine) are all built around it
- So many big companies started using it and contribute building new useful things every day (Netflix, AirBnb, Uber, Facebook, Instagram, Yahoo,)
- You might find everything you need to build application. There’re dozens of libs like: React-boostrap, react material design, react sliders, scrollers, grids, react animations and so on.

REACT COMPONENT

- Everything is/can be a component
- A React component merges view and logic
- UI is state machine,

  Components are State Processors



REACT COMPONENT

- In this training we will try to separate Containers (Smart component which fetch data, contains logic and provide logic to successors) and Dumb Components (Input data -> Output Virtual Dom description).
- Why do we need it?

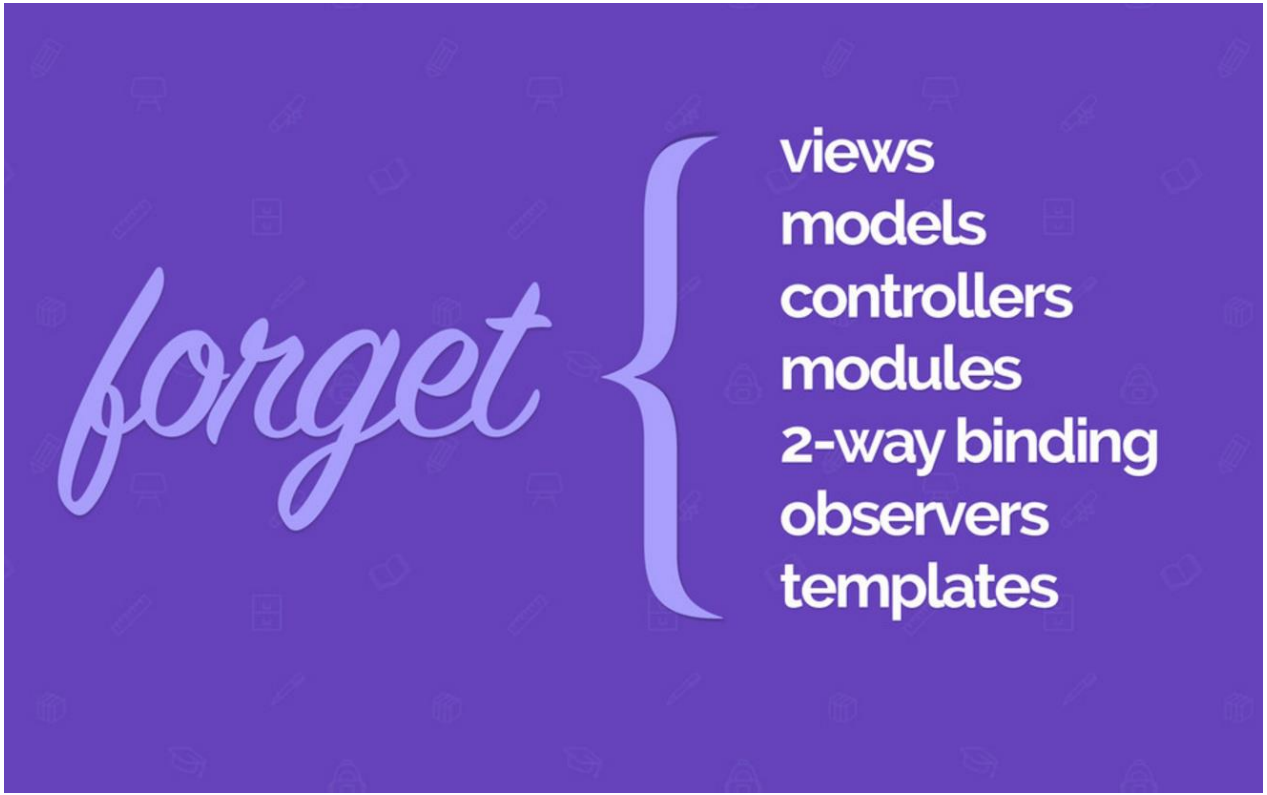
  Separates our data-fetching and rendering concerns.

  Improves reusability

  Improves validation

  This forces you to extract “layout components” such as Sidebar, Page, ContextMenu and use this.props.children instead of duplicating the same markup and layout in several container components.

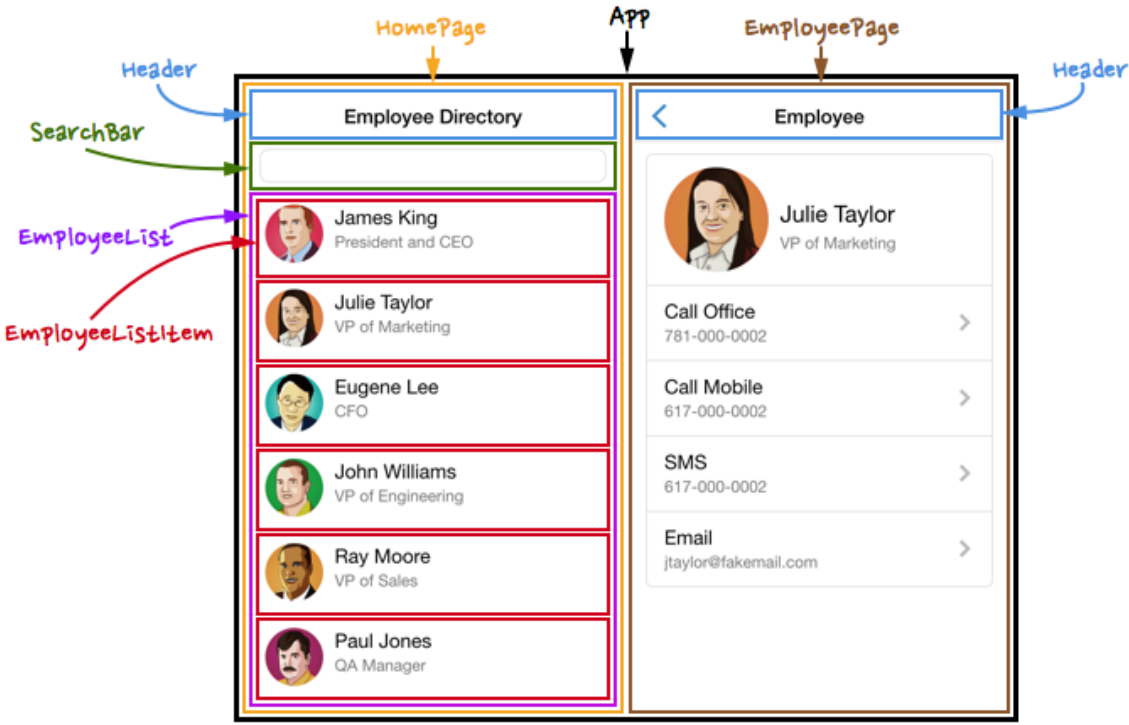
REACT COMPONENT



REACT COMPONENT

COMPONENT HIERARCHY

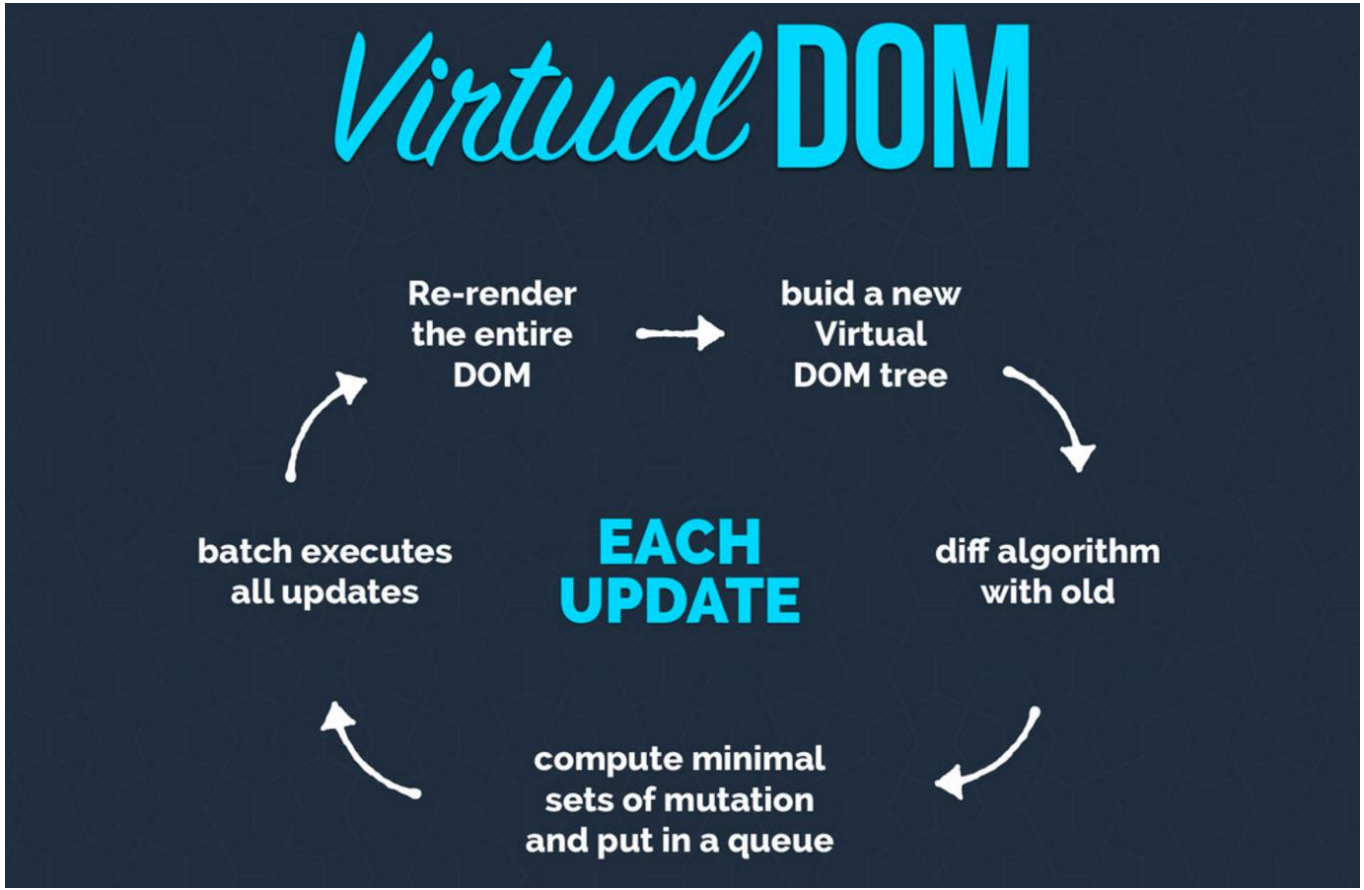
- React applications are assembled with components arranged in a hierarchy
- The easiest way to architect the application is to work out the responsibilities for each part of the interface and draw a box around it



VIRTUAL DOM

- DOM trees are huge nowadays and DOM manipulation is messy and keeping track of the previous DOM state is hard
- The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser-specific implementation details
- In React terms perhaps it's better to think of the virtual DOM as React's local and simplified copy of the HTML DOM (kinda a "mirror")
- It allows React to do its computations within this abstract world and skip the "real" DOM operations, often slow and browser-specific.

VIRTUAL DOM


Virtual DOM
Re-render the entire DOM
buid a new Virtual DOM tree
diff algorithm with old
compute minimal sets of mutation and put in a queue
batch executes all updates
EACH UPDATE

DIFF ALGORITHM

- React diff algorithm is used to compute minimum sets of mutation:
  - 1) A Span
  - 2) A Paragraph
  - 3) Remove component

None to first:

1)Create node: A Span

First to second

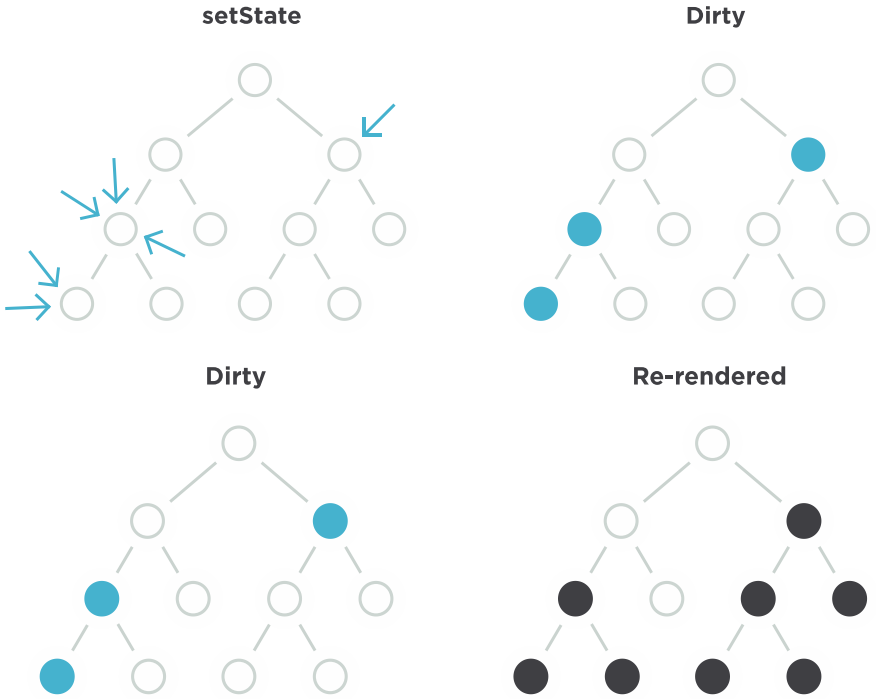
1)Replace attribute: className="first" by className="second"

2)Replace node: A Span by A Paragraph

Second to none:

1)Remove node: A Paragraph

DIFF ALGORITHM



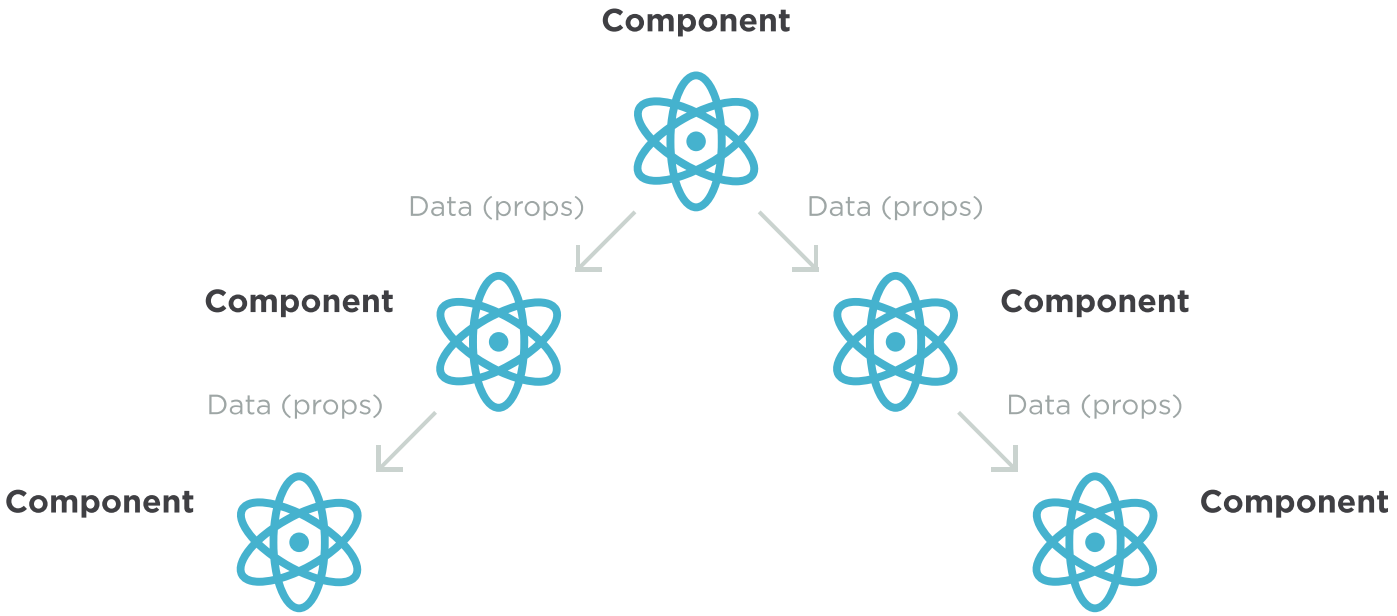
DATA FLOW

- ReactJS implements a one-way data flow. This means that data is passed from the top-down, through props, from the top component to its children, so on and so forth.

```
...
// A simple component flowing down the props
<Component property1="value" property2=”anotherValue" />
...
```

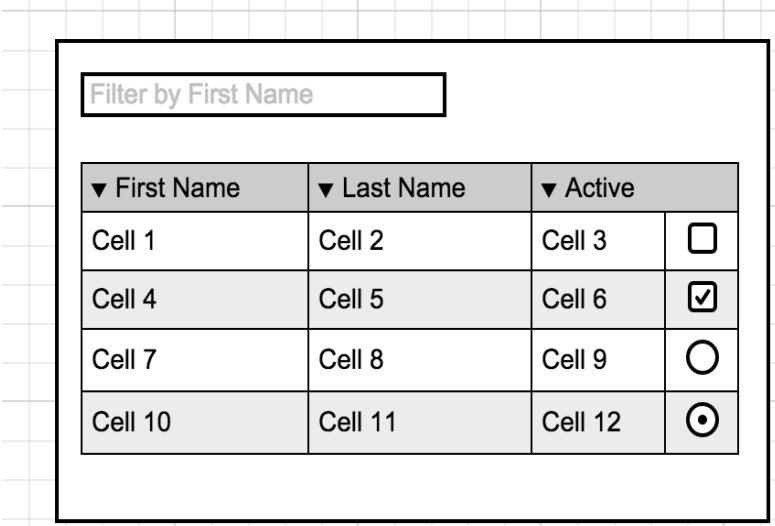
- Instead of React components talking to each other, React components cooperate through the parent

DATA FLOW


Component
Data (props)
Data (props)
Component
Component
Data (props)
Data (props)
Component
Component

DATA FLOW

- Lets say we want to develop something like this:
- First of all let separate it to the components:
  - Whole widget
  - Grid
  - Grid Record
  - Grid Action cell
  - Filter



- Components hierarchy:
  - Whole widget -> Grid -> Grid Record -> Grid Record Action
    -> Filter
- Look at this schema and you will see the only one reasonable way to build data flow in your widget.