
KEEP it
SIMPLE
EVERYTHING CAN BE A
Component

Pedro Nauck

FRONTEND DEVELOPER

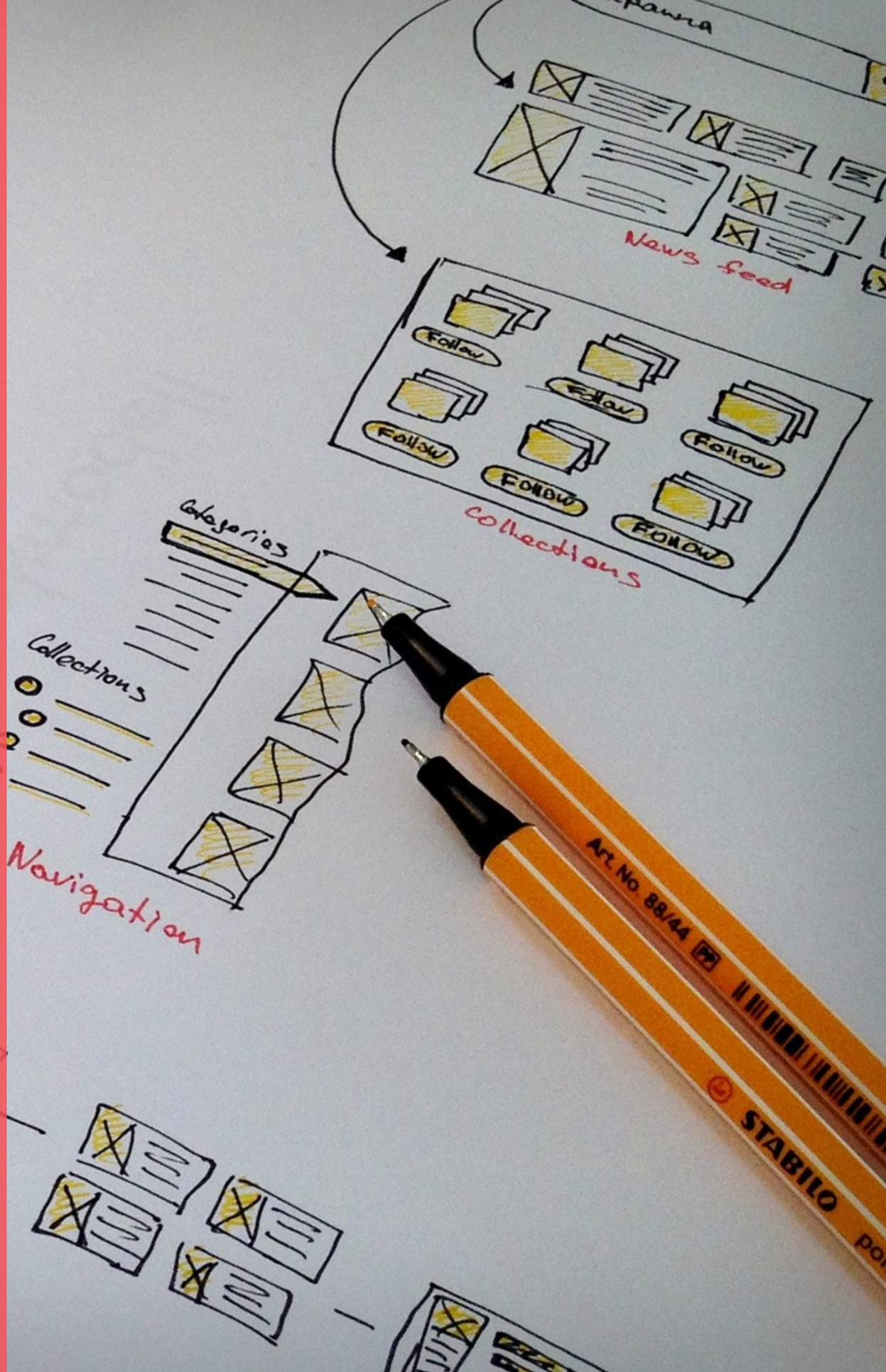
@pedronauck

pedronauck.com

USER *Interfaces*

“

The user interface is one of the **most important parts** of any program because it determines **how easily** you can make the program do what **you want.**



But, what

~~WE~~ OUR CUSTOMERS *want?*

find *Value*

we need
to deliver
that
Value


it's
not just
about
design

IT'S *about*

design
user experience
quickly adapt
performance
scalability
optimization

design
user experience
quickly adapt
performance
scalability
otimization

}

Simplicity


project size
complexity

“

Because something is simple to you, **doesn't mean** it's for everyone

when a project BECOME complex?

QUESTIONS

ANSWERS

“

Is this a module or not?

“

Do I need a directive here?

“

Should I create a factory or a service?

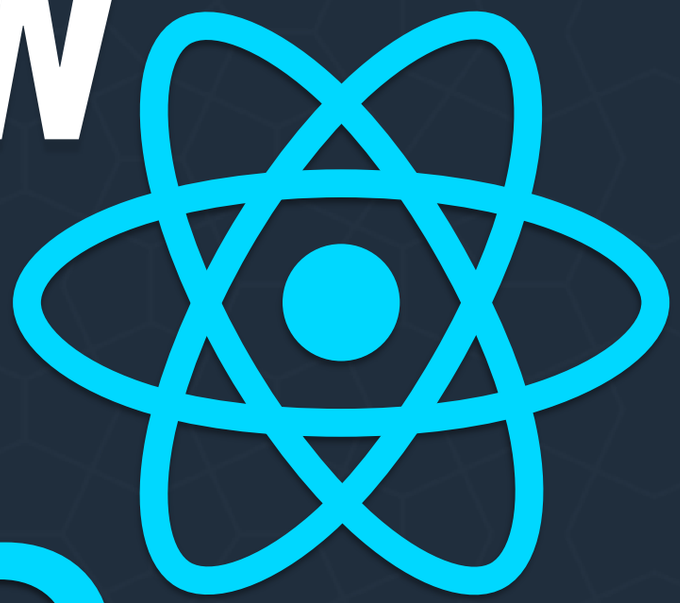
“

Why this shit doesn’t work?


Global
HD


value
complexity

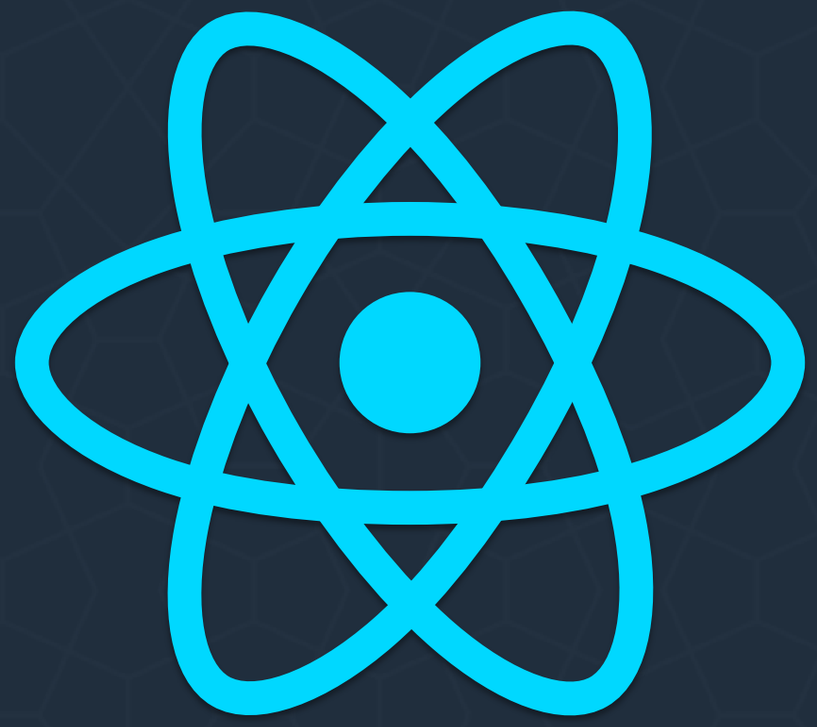
how



React

can help us

with that?



Javascript Library

for building

UI

library

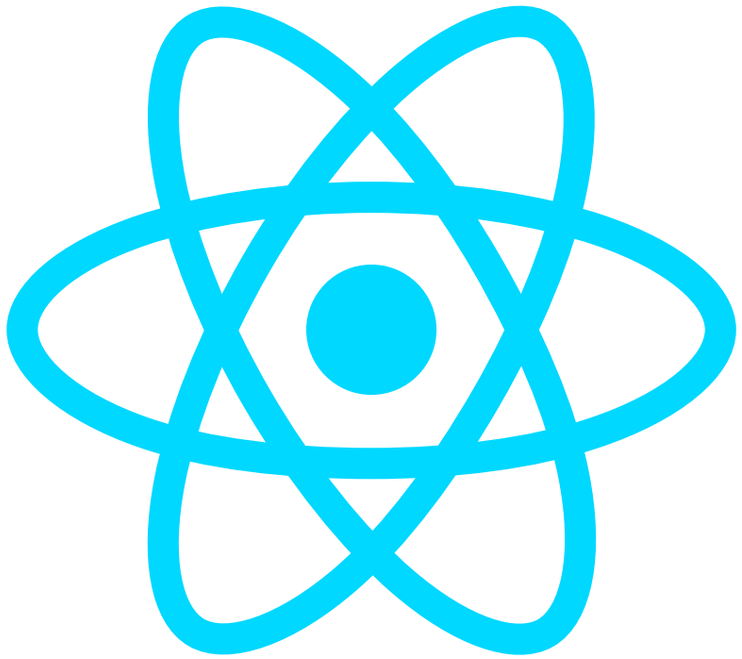
framework

library

A tool to resolve just
one specific thing

framework

A **set** of tools to
resolve a lot of things



library

framework

WHO *is using?*

facebook

KHANACADEMY

YAHOO!

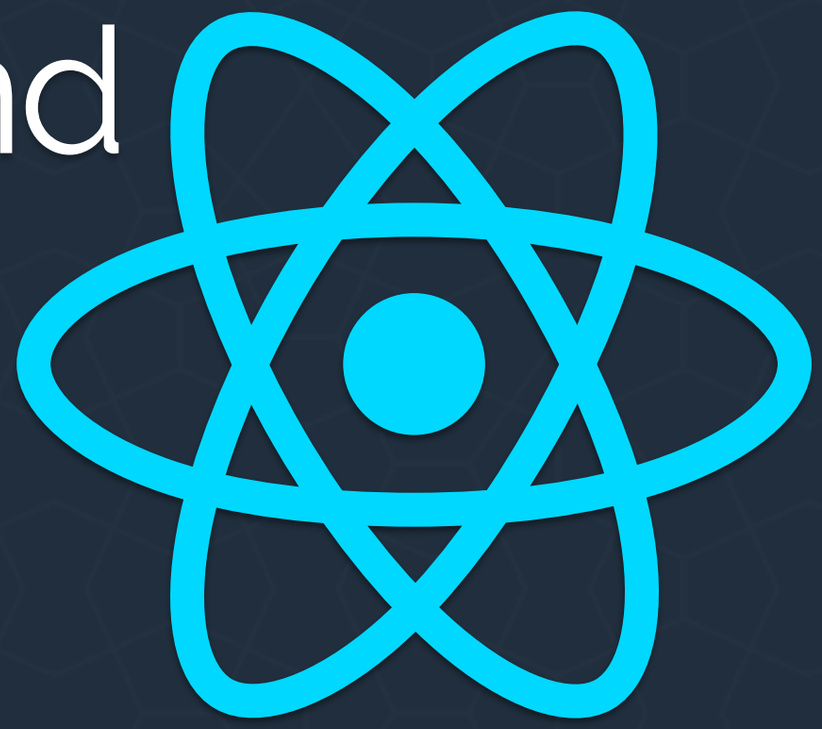
airbnb

Instagram

ATOM

THE REAL *proposal*

around



is making your

DEVELOPMENT PROCESS

simple!

HOW?


#1

WITHOUT *mutation*

WITHOUT *mutation*

We need a lightweight DOM

We need control over our data

We need to know what's happening

WE DON'T NEED


MAGIC

#2

forget

{

views
models
controllers
modules
2-way binding
observers
templates

FOCUS ON your *Components*

Component

React is all about building **reusable** components.

In fact, with React the only thing you do is build components. Since they're so **encapsulated**, components make code reuse, **testing**, and **separation of concerns** easy.

http://bit.ly/why-react

Google news

NEWS

COVERAGES

MEDIA

OPINIONS

PEOPLE

Search for coverages

MY COVERAGES FOLLOWING LATEST

+ CREATE

Coverage curated by
JIM KENNEDY

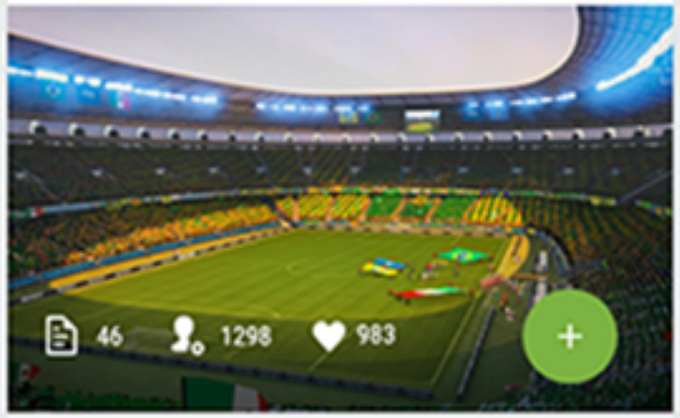
APR 23TH – JUNE 3RD

What is the core of Palestinian conflict?

Coverage curated by
KIM HJELMGAARD

JUNE 12TH – JULY 13TH

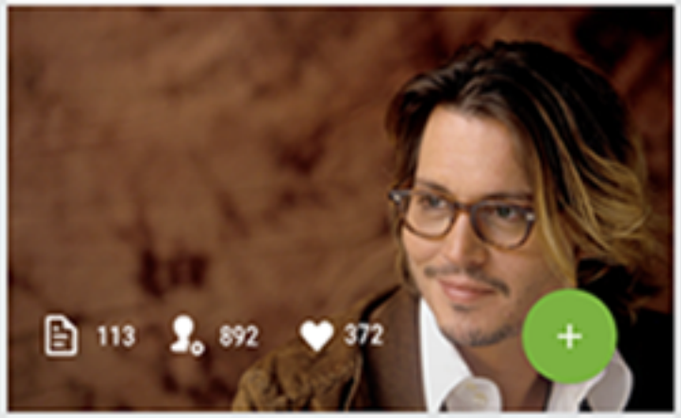
Most insteresting from World Cup 2014



Coverage curated by
CHAKER KHAZAAL and 1 more ...

JUNE 12TH 2012 – JULY 23TH 2014

Most Interesting about Johnny Depp



Coverage curated by
ALEX MUREIRA

APR 23TH – JUNE 3RD

Israel vs. Palestine: Why it's different this time

Coverage curated by
MICHAEL FAMOUS

JUNE 12TH – JULY 13TH

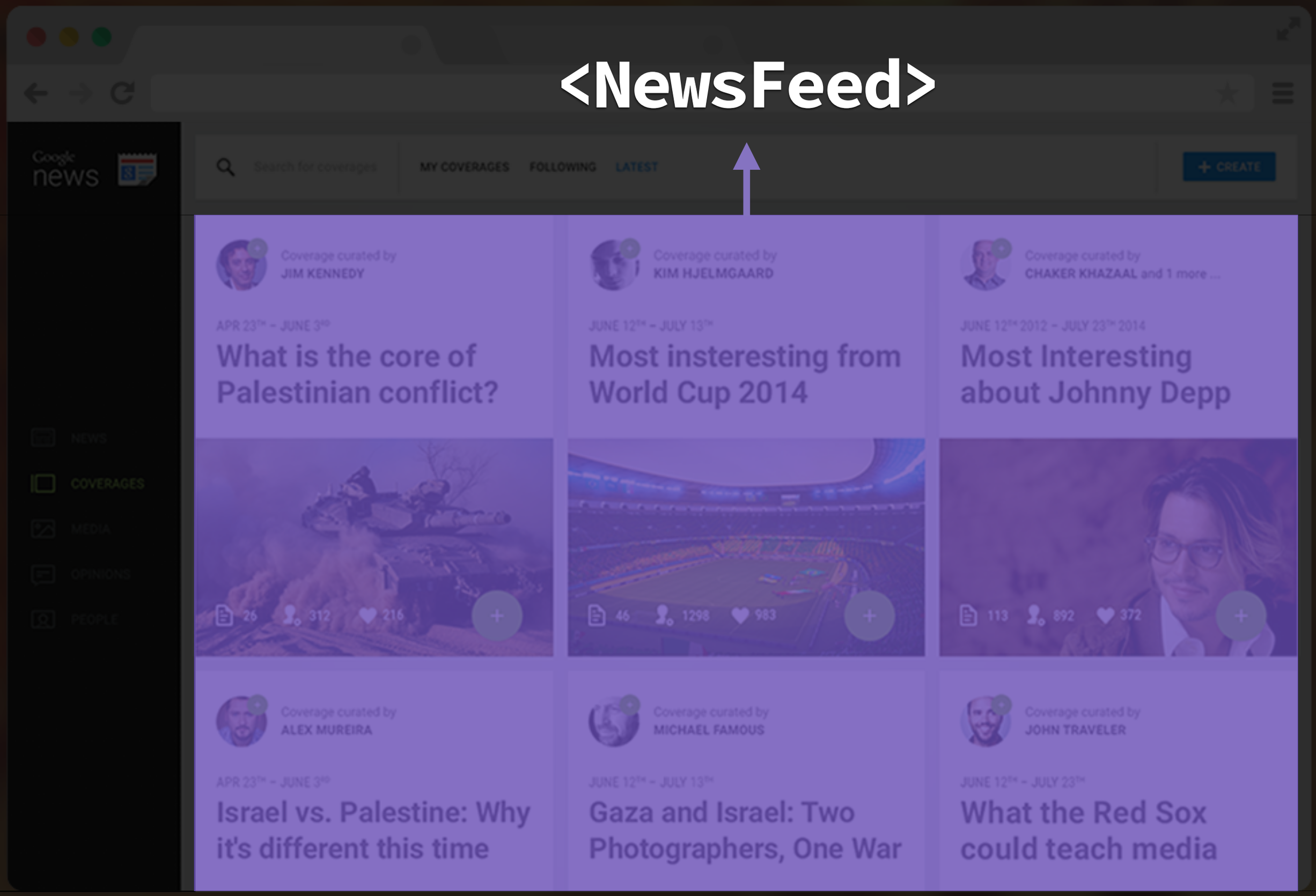
Gaza and Israel: Two Photographers, One War

Coverage curated by
JOHN TRAVELER

JUNE 12TH – JULY 23TH

What the Red Sox could teach media

<NewsFeed>


Google news
Search for coverages
MY COVERAGES
FOLLOWING
LATEST
+ CREATE
NEWS
COVERAGES
MEDIA
OPINIONS
PEOPLE
Coverage curated by
JIM KENNEDY
APR 23TH – JUNE 3RD
What is the core of Palestinian conflict?
26
312
216
Coverage curated by
KIM HJELMGAARD
JUNE 12TH – JULY 13TH
Most insteresting from World Cup 2014
46
1298
983
Coverage curated by
CHAKER KHAZAAL and 1 more ...
JUNE 12TH 2012 – JULY 23TH 2014
Most Interesting about Johnny Depp
113
892
372
Coverage curated by
ALEX MUREIRA
APR 23TH – JUNE 3RD
Israel vs. Palestine: Why it's different this time
Coverage curated by
MICHAEL FAMOUS
JUNE 12TH – JULY 13TH
Gaza and Israel: Two Photographers, One War
Coverage curated by
JOHN TRAVELER
JUNE 12TH – JULY 23TH
What the Red Sox could teach media

Google news

Search for coverages

MY COVERAGES FOLLOWING LATEST

+ CREATE

NEWS
COVERAGES
MEDIA
OPINIONS
PEOPLE

Coverage curated by
JIM KENNEDY

APR 23TH – JUNE 3RD

What is the core of Palestinian conflict?

26 312 216

Coverage curated by
KIM HJELMGAARD

JUNE 12TH – JULY 13TH

Most insteresting from World Cup 2014

46 1298 983

Coverage curated by
CHAKER KHAZAAL and 1 more ...

JUNE 12TH 2012 – JULY 23TH 2014

Most Interesting about Johnny Depp

113 892 372

<NewsItem>

Coverage curated by
ALEX MUREIRA

APR 23TH – JUNE 3RD

Israel vs. Palestine: Why it's different this time

Coverage curated by
MICHAEL FAMOUS

JUNE 12TH – JULY 13TH

Gaza and Israel: Two Photographers, One War

Coverage curated by
JOHN TRAVELER

JUNE 12TH – JULY 23TH

What the Red Sox could teach media


Google news
Search for coverages
MY COVERAGES
FOLLOWING
LATEST
+ CREATE
COVERAGES
MEDIA
OPINIONS
PEOPLE
Coverage curated by
JIM KENNEDY
APR 23TH – JUNE 3RD
What is the core of Palestinian conflict?
<NewsAuthor>
<NewsTitle>
<NewsCover>
<NewsStats>
Coverage curated by
ALEX MUREIRA
APR 23TH – JUNE 3RD
Israel vs. Palestine: Why it's different this time
Coverage curated by
MICHAEL FAMOUS
JUNE 12TH – JULY 13TH
Gaza and Israel: Two Photographers, One War
Coverage curated by
JOHN TRAVELER
JUNE 12TH – JULY 23TH
What the Red Sox could teach media

THIS IS just *the concept*

SHOW ME

some code

please!

NewsItem.jsx

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.jsx

factory
pattern

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

JSX

A Javascript XML based extension that compile **just the template** part of your code

“

We strongly believe that components are the right way to separate concerns rather than *"templates"* and *"display logic."*

We think that **markup** and the **code that generates it** are **intimately tied together.**

facebook

“

It makes a lot of sense that a JavaScript's template **stays** in the JavaScript

@jcemer

#1 WHAT'S THE *difference?*

the concerns are **mixing too**

```
<article className='news-item'>
  <div className='news-author'>
    <%= @post.author.name %>
  </div>
  <header className='news-title'>
    <%= @post.title %>
  </header>
  <div className='news-cover'>
    <%= image_tag @post.image %>
  </div>
</article>
```

#2 TO MUCH BETTER *than*

```
var title = 'Some title';
var author = 'Jim Keneddy';
var newsItem =  '<article class="news-item">' +
                  '<div class="news-author">' + author + '</div>' +
                  '<header class="news-title">' + title + '</header>' +
                  '<div class="news-cover">' + '...' + '</div>' +
                  '<ul class="news-stats"> + '...' + </ul>' +
                '</article>';
```

NewsItem.jsx

#3 WILL BE *Compiled*

```
var NewsItem = React.createClass({
  render: function() {
    return (
      <article className='news-item'>
        <div className='news-author'>...</div>
        <header className='news-title'>...</header>
        <figure className='news-cover'>...</figure>
        <ul className='news-stats'>...</ul>
      </article>
    );
  }
});
```

NewsItem.js

```
var React = require('react');

var NewsItem = React.createClass({
  render: function() {
    return (
      React.createElement('article', {className: 'news-item'},
        React.createElement('div', {className: 'news-author'}, '...'),
        React.createElement('header', {className: 'news-title'}, '...'),
        React.createElement('figure', {className: 'news-cover'},
          '...',
          React.createElement('ul', {className: 'news-stats'}, '...')
        )
      )
    );
  }
});
```

BUT

Why?

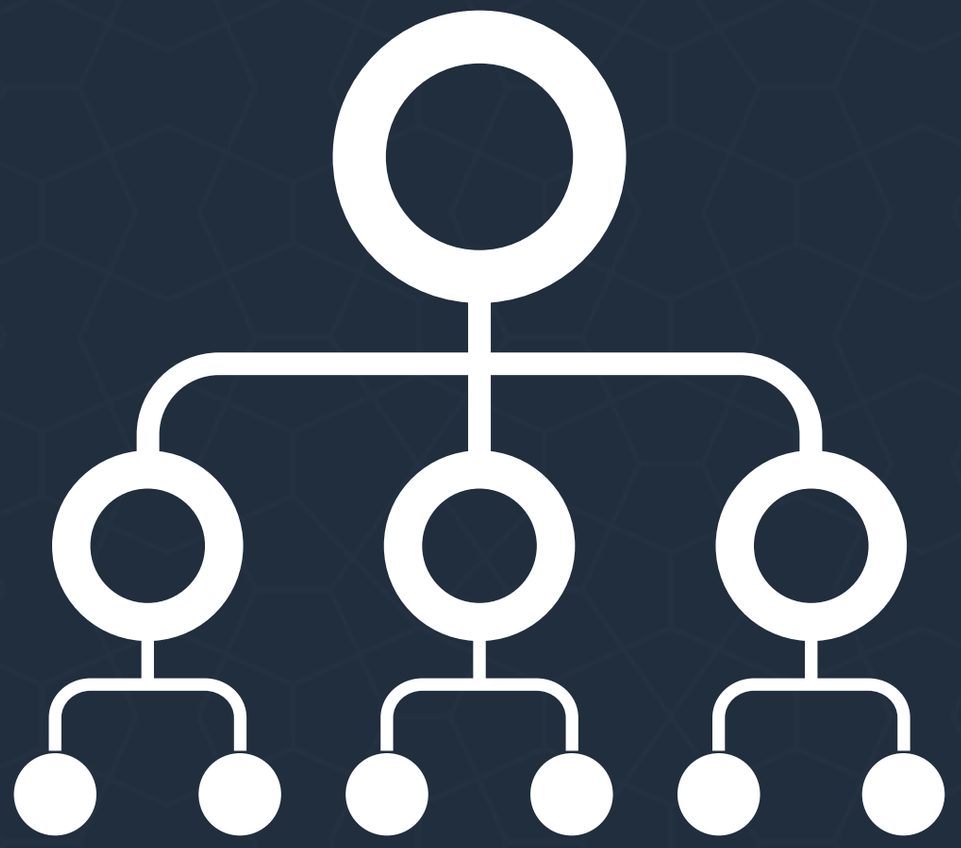
BECAUSE

mutation

is EVIL

Virtual

DOM

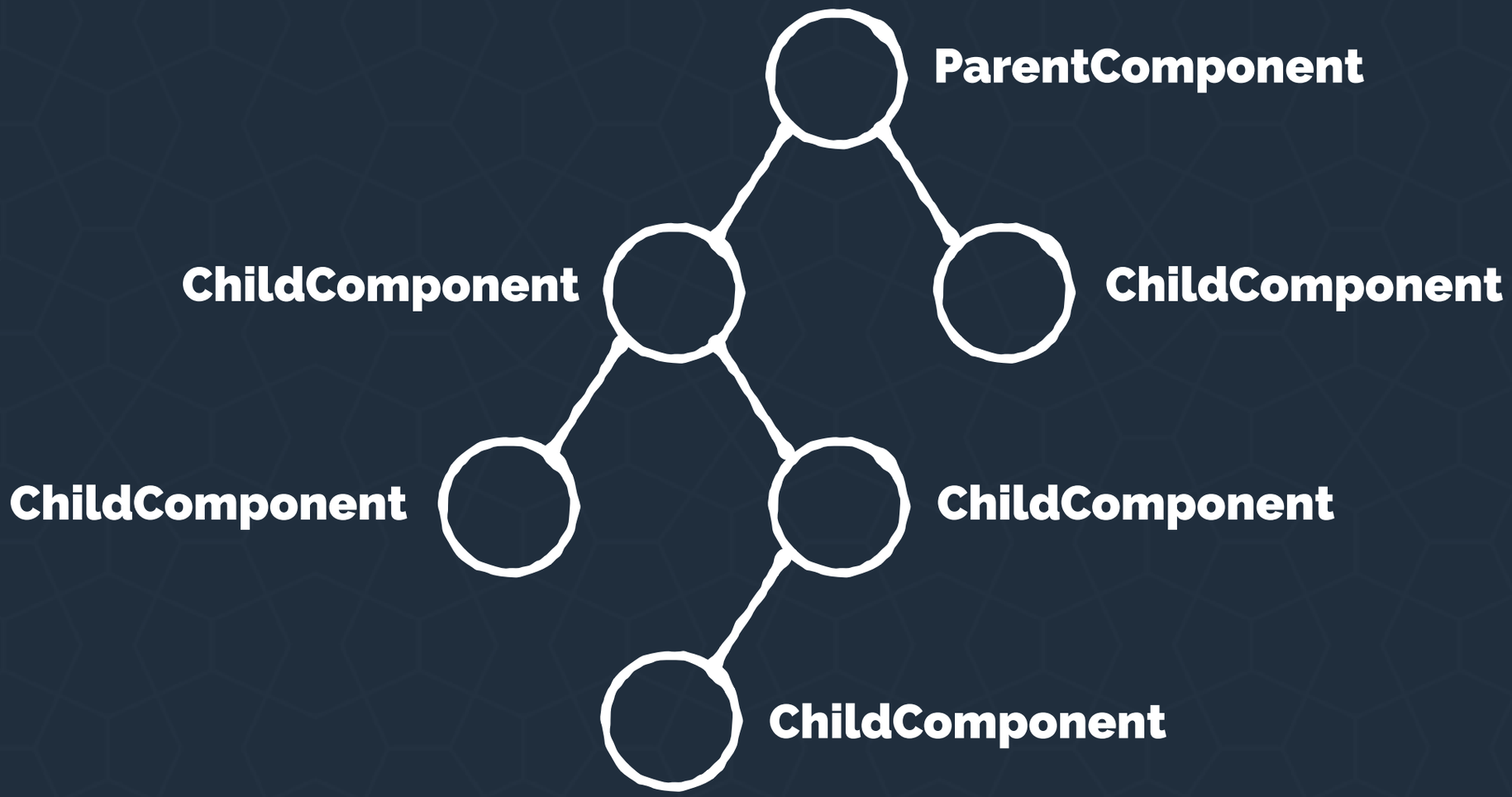


Virtual DOM

An object tree that is a
representation of the DOM.
Kinda a **"mirror".**

DOM
Travolta

Virtual DOM


ParentComponent
ChildComponent
ChildComponent
ChildComponent
ChildComponent
ChildComponent

Virtual DOM

buid a new
Virtual
DOM tree

diff algorithm
with old

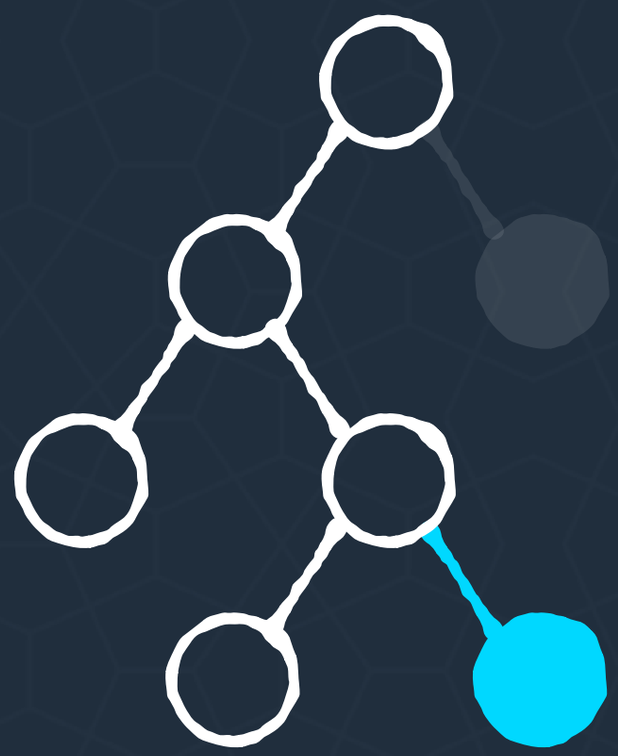
compute minimal
sets of mutation
and put in a queue

batch executes
all updates

EACH UPDATE

Virtual DOM

Old Tree



Mutations

New Tree

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

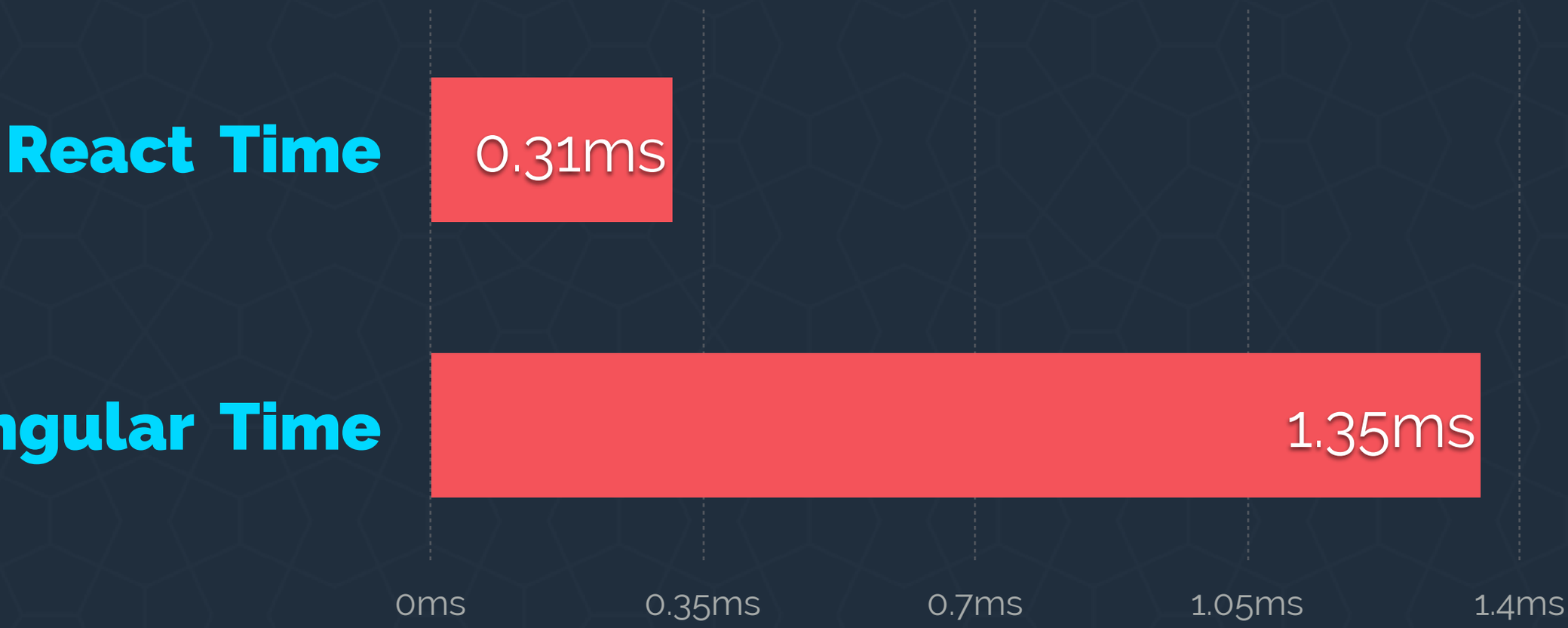
ADVANTAGES:

Javascript is Fast

and the DOM **is slow**

Benchmark

Render a list with **1500 rows**




Fascinating.

NewsItem.jsx

kinda messy!

```
render: function() {
  return (
    <article className='news-item'>
      <div className='news-author'>...</div>
      <header className='news-title'>...</header>
      <figure className='news-cover'>
        ...
        <ul className='news-stats'>...</ul>
      </figure>
    </article>
  );
}
```

WE NEED to *break*

Props & State

Props & STATE

Where you **manage the data**
of your components

Props

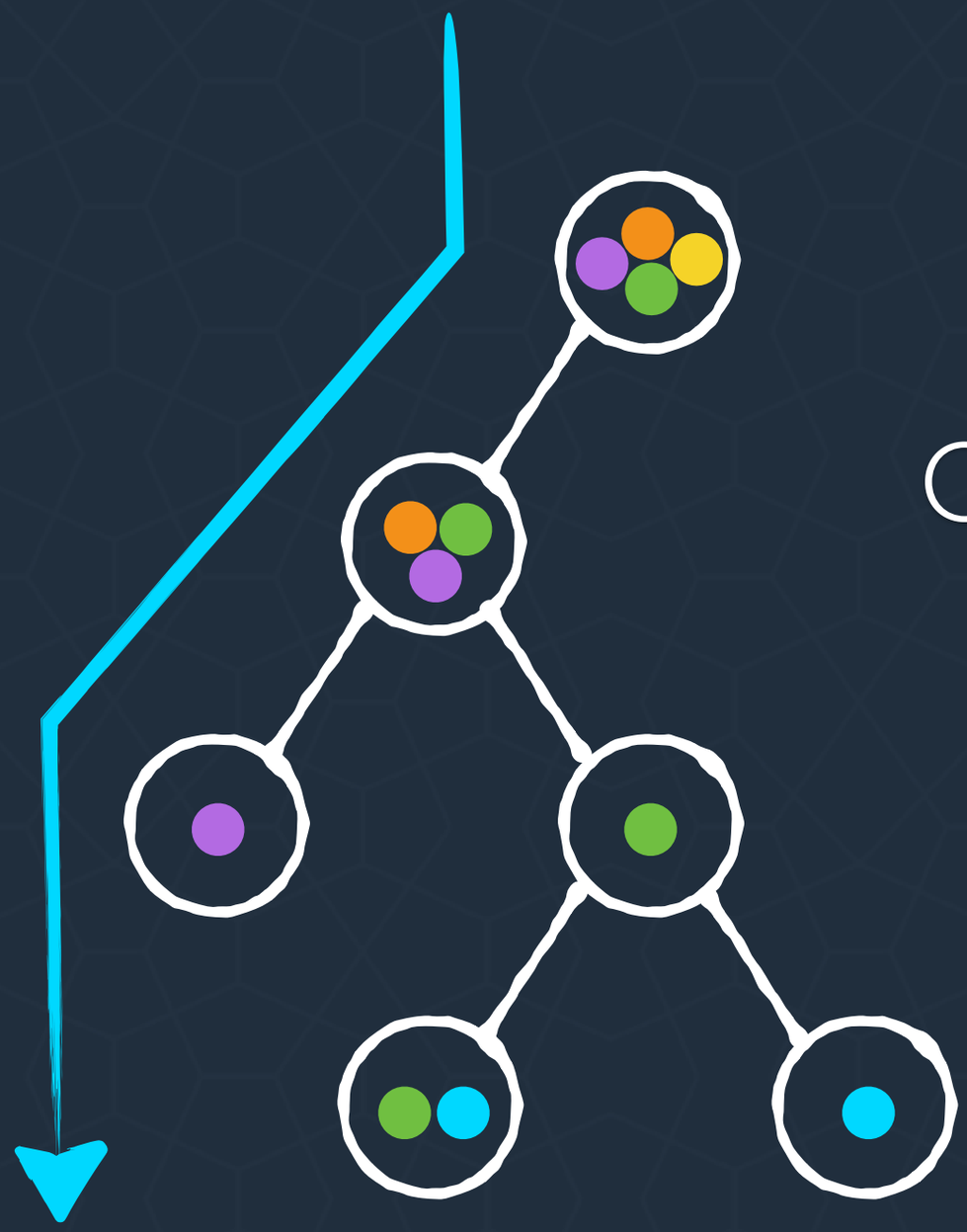
Data that can be
imutable

State

Data that can be
mutable

Unidirectional

DATAFLOW



Components can either get imutable data **(via props)** or management their own state **(via state)**

NewsFeed.jsx

```jsx
var React = require('react');
var NewsItem = require('./components/NewsItem');

var NewsFeed = React.createClass({
  render: function() {
    var posts = this.props.posts.map(function(post) {
      return <NewsItem post={post} key={post._id} />
    });

    return <div className='news-feed'>{posts}</div>
  }
});
```

NewsFeed.jsx

```
var React = require('react');
var NewsItem = require('./components/NewsItem');

var NewsFeed = React.createClass({
  render: function() {
    var posts = this.props.posts.map(function(post) {
      return <NewsItem post={post} key={post._id} />
    });

    return <div className='news-feed'>{posts}</div>
  }
});
```

NewsFeed.jsx

```
var React = require('react');
var NewsItem = require('./components/NewsItem');

var NewsFeed = React.createClass({
  render: function() {
    var posts = this.props.posts.map(function(post) {
      return <NewsItem post={post} key={post._id} />
    });

    return <div className='news-feed'>{posts}</div>
  }
});
```

NewsItem.jsx

```jsx
var React = require('react');
var NewsAuthor = require('./component/NewsAuthor');
var NewsTitle = require('./component/NewsTitle');
var NewsCover = require('./component/NewsCover');
var NewsStats = require('./component/NewsStats');

var NewsItem = React.createClass({
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item'>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');
var NewsAuthor = require('./component/NewsAuthor');
var NewsTitle = require('./component/NewsTitle');
var NewsCover = require('./component/NewsCover');
var NewsStats = require('./component/NewsStats');

var NewsItem = React.createClass({
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item'>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

```
var React = require('react');
var NewsAuthor = require('./component/NewsAuthor');
var NewsTitle = require('./component/NewsTitle');
var NewsCover = require('./component/NewsCover');
var NewsStats = require('./component/NewsStats');

var NewsItem = React.createClass({
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item'>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

received data
from **NewsFeed**

NewsItem.jsx

breaking data to
each component
via props

```
<NewsAuthor author={post.author} />
<NewsTitle title={post.title} />
<NewsCover image={post.cover} />
<NewsStats postStats={post.stats} />
```

NewsTitle.jsx

```
var NewsTitle = React.createClass({
  render: function() {
    var createdAt = this.props.title.createdAt;
    var value = this.props.title.value;

    return (
      <header className='news-title'>
        <span className='news-title-data'>{createdAt}</span>
        <h2 className='news-title-value'>{value}</h2>
      </header>
    );
  }
});
```

NewsTitle.jsx

```
var NewsTitle = React.createClass({
  render: function() {
    var createdAt = this.props.title.createdAt;
    var value = this.props.title.value;

    return (
      <header className='news-title'>
        <span className='news-title-data'>{createdAt}</span>
        <h2 className='news-title-value'>{value}</h2>
      </header>
    );
  }
});
```

your

browser

WILL RENDER...

etwork Sources Timeline Profiles Resources Audits Console

```
<!DOCTYPE html>
▼ <html>
  ▶ <head>…</head>
  ▼ <body>
    ▼ <div id="content">
      ▼ <div class="news-feed" data-reactid=".0">
        ▼ <article class="news-item" data-reactid=".0.$0">
          ▶ <div class="news-author" data-reactid=".0.$0.0">…</div>
          ▶ <header class="news-title" data-reactid=".0.$0.1">…</header>
          ▶ <figure class="news-cover" data-reactid=".0.$0.2">…</figure>
          ▶ <ul class="news-stats" data-reactid=".0.$0.3">…</ul>
          </article>
        ▶ <article class="news-item" data-reactid=".0.$1">…</article>
        ▶ <article class="news-item" data-reactid=".0.$2">…</article>
        ▶ <article class="news-item" data-reactid=".0.$3">…</article>
      </div>
</div>
```

keeping a **good**

semantic

Validating PROPS

NewsTitle.jsx

```
var NewsTitle = React.createClass({
  propTypes: {
    title: React.PropTypes.object.isRequired
  },
  render: function() {
    var createdAt = this.props.title.createdAt;
    var value = this.props.title.value;

    return (
      <header className='news-title'>
        <span className='news-title-data'>{createdAt}</span>
        <h2 className='news-title-value'>{value}</h2>
      </header>
    );
  }
});
```

NewsTitle.jsx

if is a **required** property

```
var NewsTitle = React.createClass({
  propTypes: {
    title: React.PropTypes.object.isRequired
  },
  render: function() {
    var createdAt = this.props.title.createdAt;
    var value = this.props.title.value;
    return (
      <header className='news-title'>
        <span className='news-title-data'>{createdAt}</span>
        <h2 className='news-title-value'>{value}</h2>
      </header>
    );
  }
});
```

property **name**

property **type**

```
propTypes: {
  optionalArray: React.PropTypes.array,
  optionalBool: React.PropTypes.bool,
  optionalFunc: React.PropTypes.func,
  optionalNumber: React.PropTypes.number,
  optionalObject: React.PropTypes.object,
  optionalString: React.PropTypes.string,
  ...
},
```

a bunch of types and options

http://bit.ly/reusable-components

USEFULL *for*

Generate a **manifest** of your component

Throw **exceptions** when passed a invalid prop

and **about**

States

NewsItem.jsx

```jsx
var NewsItem = React.createClass({
  getInitialState: function() {
    return { isBookmarked: false };
  },
  handleSelect: function(e) {
    this.setState({ isBookmarked: true });
    e.preventDefault();
  },
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item' onClick={this.handleSelect}>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

```
var NewsItem = React.createClass({
  getInitialState: function() {
    return { isBookmarked: false };
  },
  handleSelect: function(e) {
    this.setState({ isBookmarked: true });
    e.preventDefault();
  },
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item' onClick={this.handleSelect}>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

```
var NewsItem = React.createClass({
  getInitialState: function() {
    return { isBookmarked: false };
  },
  handleSelect: function(e) {
    this.setState({ isBookmarked: true });
    e.preventDefault();
  },
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item' onClick={this.handleSelect}>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

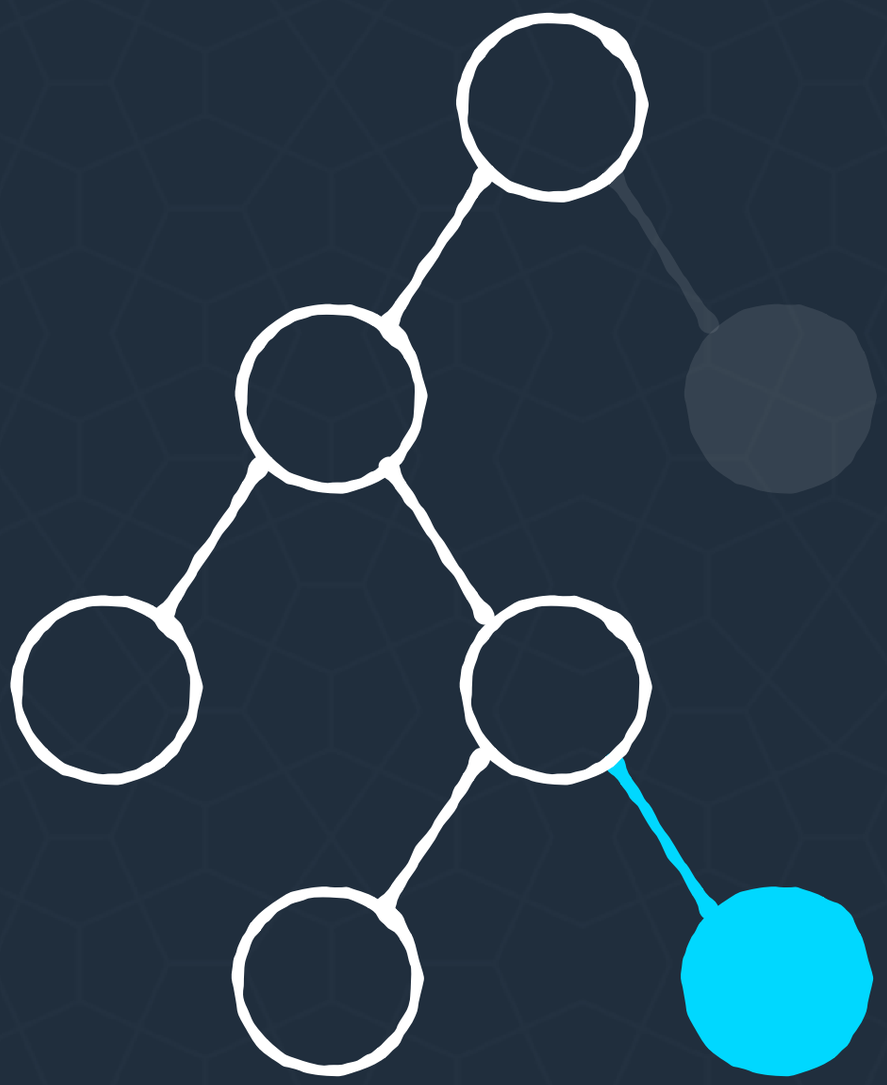
```
var NewsItem = React.createClass({
  getInitialState: function() {
    return { isBookmarked: false };
  },
  handleSelect: function(e) {
    this.setState({ isBookmarked: true });
    e.preventDefault();
  },
  render: function() {
    var post = this.props.post;

    return (
      <article className='news-item' onClick={this.handleSelect}>
        <NewsAuthor author={post.author} />
        <NewsTitle title={post.title} />
        <NewsCover image={post.cover} />
        <NewsStats postStats={post.stats} />
      </article>
    );
  }
});
```

NewsItem.jsx

```
getInitialState: function() {
  return { isBookmarked: false };
},
handleSelect: function(e) {
  this.setState({ isBookmarked: true });
  e.preventDefault();
},
```

managing state ✓

```
onClick={this.handleSelect}
```



Virtual DOM

will re-render **on each**

```
this.setState();
```

main.jsx

```
var React = require('react');
var NewsFeed = require('./components/NewsFeed');
var posts = $.ajax('GET', '/some/api/url/post');

posts.success(function(response) {
  React.render(
    <NewsFeed posts={response.body.posts} />,
    document.getElementById('#myDomNode')
  );
});
```

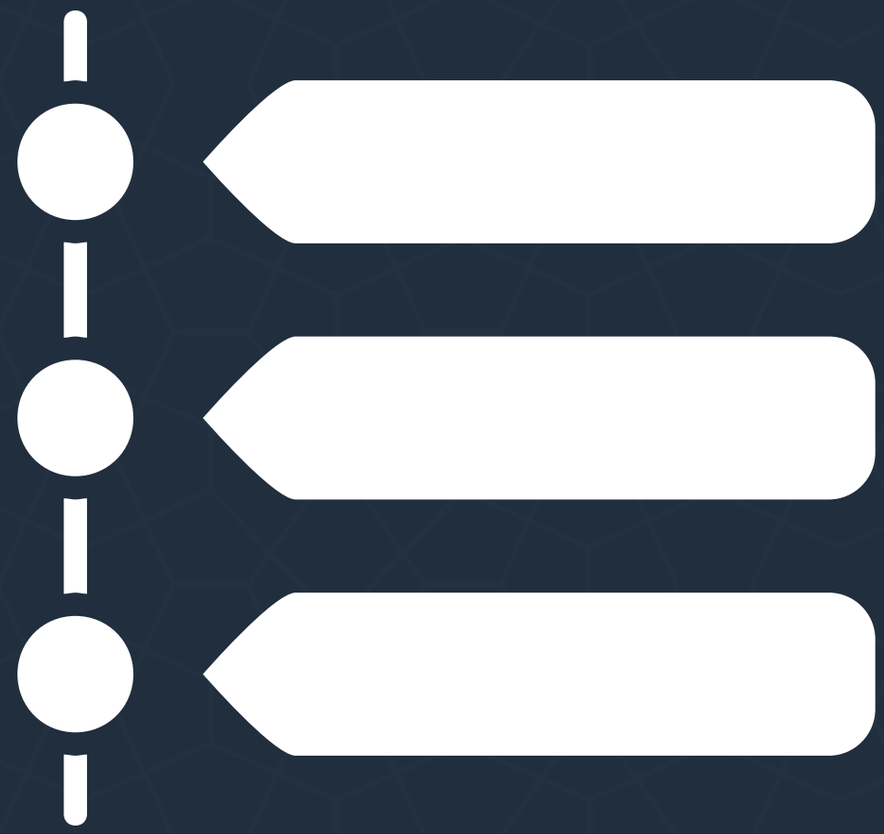
main.jsx

```
var React = require('react');
var NewsFeed = require('./components/NewsFeed');
var posts = $.ajax('GET', '/some/api/url/post');
posts.success(function(response) {
  React.render(
    <NewsFeed posts={response.body.posts} />,
    document.getElementById('#myDomNode')
  );
});
```

main.jsx

```
var React = require('react');
var NewsFeed = require('./components/NewsFeed');
var posts = $.ajax('GET', '/some/api/url/post');

posts.success(function(response) {
  React.render(
    <NewsFeed posts={response.body.posts} />,
    document.getElementById('#myDomNode')
  );
});
```

component LIFE Cycle



component LIFE*Cycle*

Callbacks that become possible to interact with Virtual DOM and **manipulate** the real DOM

MOUNTING

componentWillMount()
componentDidMount()

UNMOUNTING

componentWillUnmount()

UPDATING

componentWillReceiveProps(nextProps)
shouldComponentUpdate(nextProps, nextState)
componentWillUpdate(nextProps, nextState)
componentDidUpdate(prevProps, prevState)

USEFULL *for*

Manage virtual dom render updates

Manage **state** of your component

Integrate **third-party** scripts

Slider.jsx

```
var Slider = React.createClass({
  render: function() {
    return (
      <ul className='slider'>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
      </ul>
    );
  },
  componentDidMount: function () {
    var sliderNode = this.getDOMNode();
    $(sliderNode).sliderPlugin();
  }
});
```

Slider.jsx

```
var Slider = React.createClass({
  render: function() {
    return (
      <ul className='slider'>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
      </ul>
    );
  },
  componentDidMount: function () {
    var sliderNode = this.getDOMNode();
    $(sliderNode).sliderPlugin();
  }
});
```

Slider.jsx

```
var Slider = React.createClass({
  render: function() {
    return (
      <ul className='slider'>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
      </ul>
    );
  },
  componentDidMount: function () {
    var sliderNode = this.getDOMNode();
    $(sliderNode).sliderPlugin();
  }
});
```

Slider.jsx

```
var Slider = React.createClass({
  render: function() {
    return (
      <ul className='slider'>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
        <li className='slider-items'>...</li>
      </ul>
    );
  },
  componentDidMount: function () {
    var sliderNode = this.getDOMNode();
    $(sliderNode).sliderPlugin();
  }
});
```

Mixins

Mixins

Snippets of **reusable code** that can be injected on your component **without modifying** it's main behavior

NewsStats.jsx

```
var NewsStats = React.createClass({
  render: function() {
    // some stuff
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
});

var NewsAuthor = React.createClass({
  render: function() {
    // some stuff
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
});
```

NewsStats.jsx

```
var NewsStats = React.createClass({
  render: function() {
    // some stuff
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
});
```

we need to
be **DRY** here

```
var NewsAuthor = React.createClass({
  render: function() {
    // some stuff
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
});
```

AuthMixin.jsx

```
var AuthMixin = {
  getInitialState: function() {
    return { isAuth: false };
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
};
```

AuthMixin.jsx

```
var AuthMixin = {
  getInitialState: function() {
    return { isAuth: false };
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
};
```

AuthMixin.jsx

```
var AuthMixin = {
  getInitialState: function() {
    return { isAuth: false };
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
};
```

AuthMixin.jsx

```
var AuthMixin = {
  getInitialState: function() {
    return { isAuth: false };
  },
  checkIfAuthenticated: function() {
    if (MyData.isAuthenticated()) {
      this.setState({ isAuth: true });
    }
  }
};
```

NewsAuthor.jsx

```
var AuthMixin = require('./mixins/AuthMixin');

var NewsAuthor = React.createClass({
  mixins: [AuthMixin],
  render: function() {
    this.checkIfAuthenticated();
  }
});
```

NewsAuthor.jsx

```
var AuthMixin = require('./mixins/AuthMixin');

var NewsAuthor = React.createClass({
  mixins: [AuthMixin],
  render: function() {
    this.checkIfAuthenticated();
  }
});
```

NewsAuthor.jsx

```
var AuthMixin = require('./mixins/AuthMixin');

var NewsAuthor = React.createClass({
  mixins: [AuthMixin],
  render: function() {
    this.checkIfAuthenticated();
  }
});
```

NewsAuthor.jsx

```
var AuthMixin = require('./mixins/AuthMixin');

var NewsAuthor = React.createClass({
  mixins: [AuthMixin],
  render: function() {
    this.checkIfAuthenticated();
  }
});
```

SERVER *Render*

SERVERRender

Render your component **both on server and client**, making your application **Isomorphic** and and **crawlable**.

SPA

Server

Persistence via API

Client

Form Validation

DOM Manipulation

Animations

View layer

Rounting/Controller

Application Logic

IsomorphicSPA

Server

Persistence via API

Client

Form Validation

DOM Manipulation

Animations

View layer

Rounting/Controller

Application Logic

Render

Server

Render **static**
markup
via server

Client

Attach
components
behavior

server.js

```
var React = require('react');
var NewsFeed = require('./components/NewFeed');
var posts = api.fetchPosts();

var html = React.renderToString(
  <NewsFeed posts={posts} />
);

server.renderHtml(html);
```

server.js

```
var React = require('react');
var NewsFeed = require('./components/NewFeed');
var posts = api.fetchPosts();

var html = React.renderToString(
  <NewsFeed posts={posts} />
);

server.renderHtml(html);
```

server.js

```
var React = require('react');
var NewsFeed = require('./components/NewFeed');
var posts = api.fetchPosts();

var html = React.renderToString(
  <NewsFeed posts={posts} />
);

server.renderHtml(html);
```

server.js

```
var React = require('react');
var NewsFeed = require('./components/NewFeed');
var posts = api.fetchPosts();

var html = React.renderToString(
  <NewsFeed posts={posts} />
);

server.renderHtml(html); // just to illustrate
```

in a *nutshell*

React introduces a new way to build **scalable** applications that can take you out of your ***comfort zone***

in a *nutshell*

You need to ***clean your mind***
about a lot of **old "trues"**

in a *nutshell*

It's about **deliver value**.
If something help you to make that.
Why do you worry about write your
HTML inside Javascript?

I hope you *enjoyed*

Almir Filho @almirfilho · Oct 20

Será possível, que toda palestra sobre ReactJS seja chata? Já vi umas 5 e todas dão imenso sono.

Pedro Nauck

FRONTEND DEVELOPER

@pedronauck

pedronauck.com

Questions?