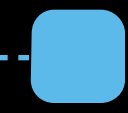
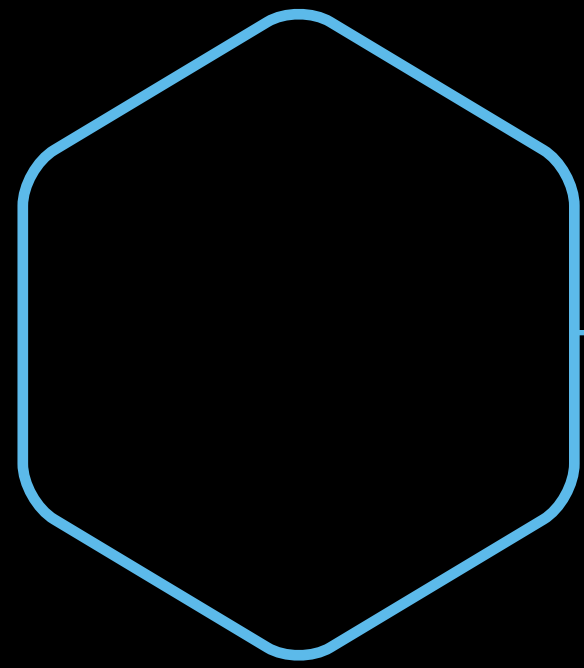


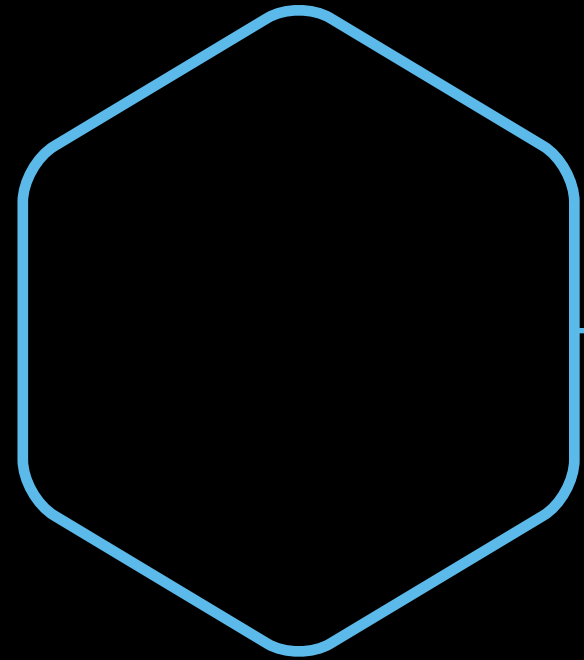
# BUILDING FAST WEB UI WITH REACT

Pratik Patel @prpatel

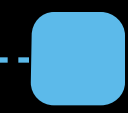
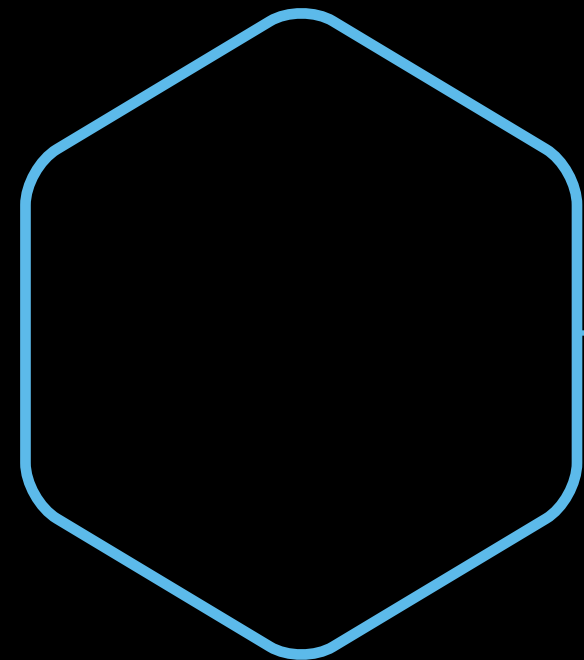
@prpatel



**UI ONLY LIBRARY**



**DEVELOPED AT FACEBOOK**

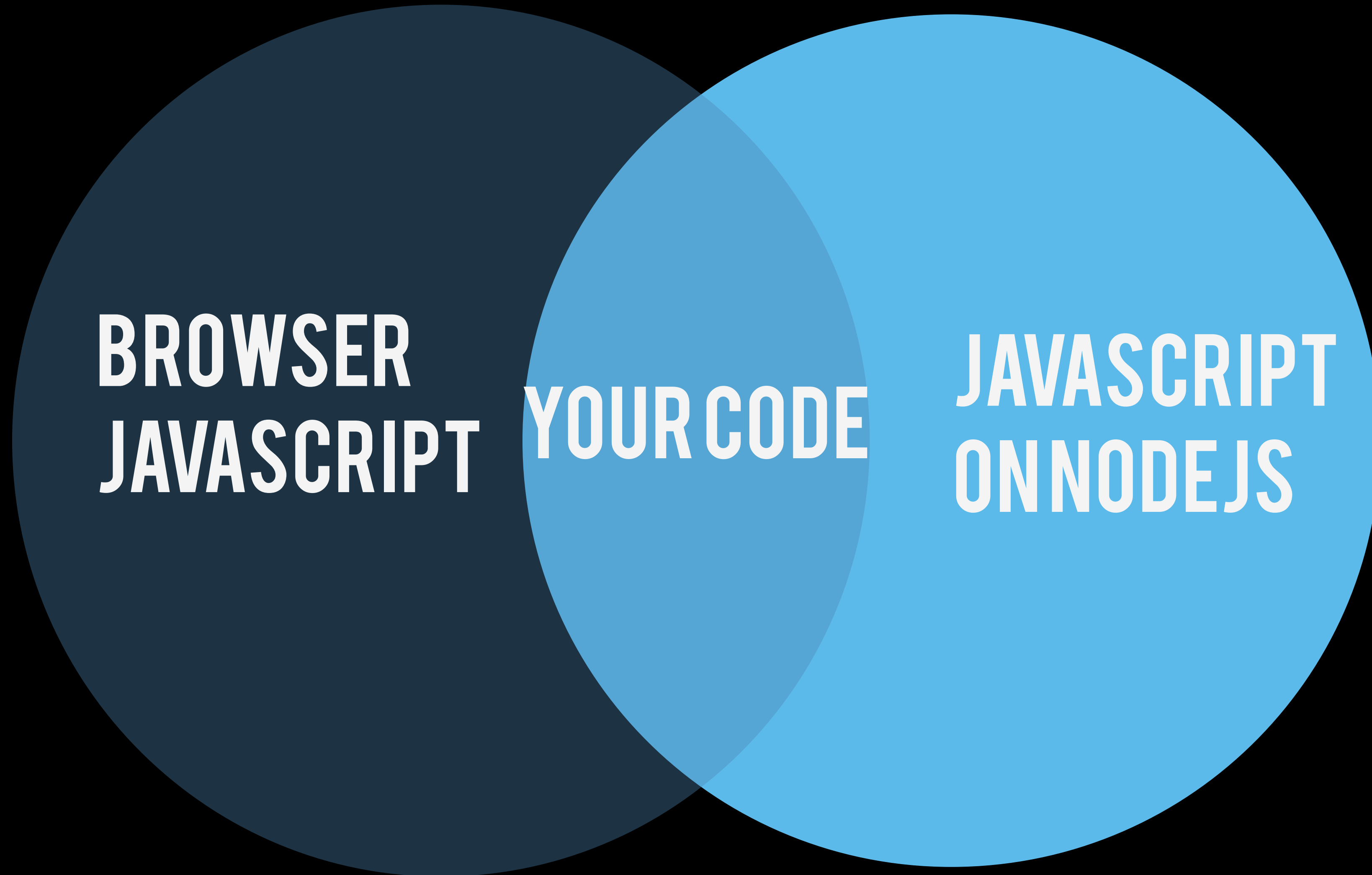


**POWERS INSTAGRAM.COM**

# REACTJS CONCEPTS

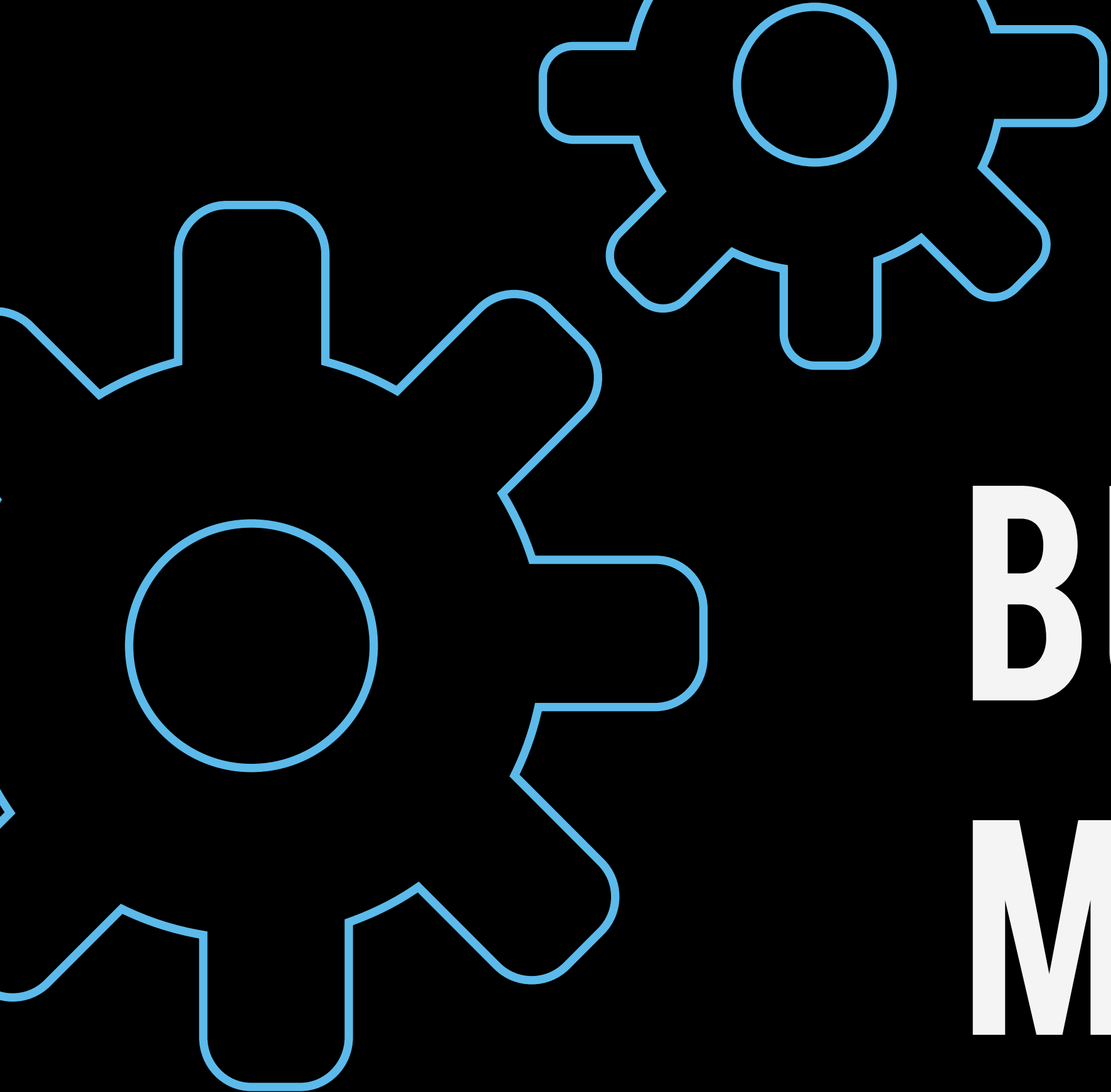
**CODE RUNS ON THE BROWSER  
\_OR\_  
THE SERVER**

**ISOMORPHIC**

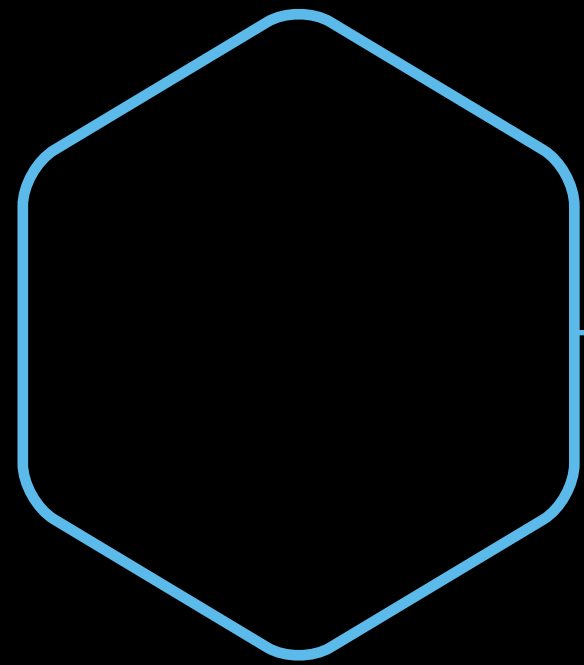




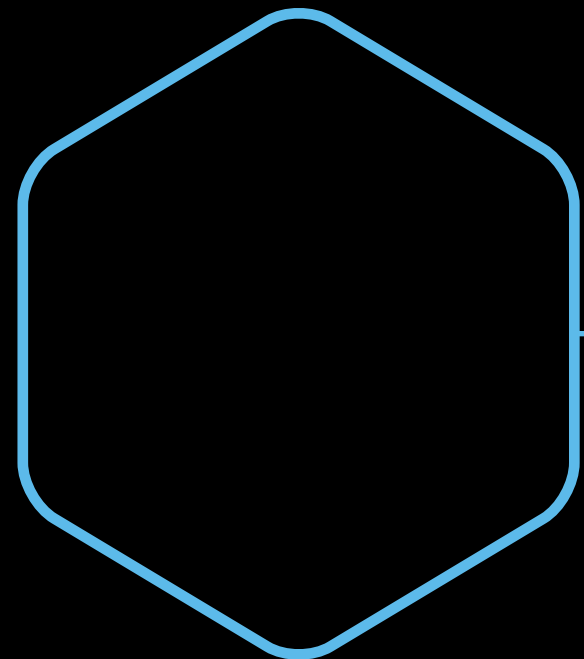
**Allows great  
flexibility and  
performance  
management**



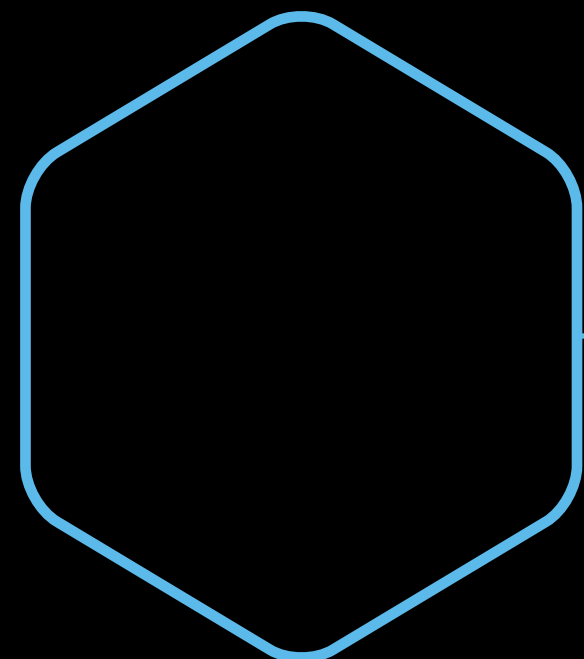
**BUT THAT'S NOT THE  
MAIN REASON  
REACTJS IS FAST**



**VIRTUAL DOM**



**SMART DIFF'ING OF DOM**



**BATCHED DOM UPDATES**





# TYPICAL BROWSER UPDATE

**PARSE  
MARKUP  
&  
CSS**

**CREATE  
RENDER  
TREE**

**REFLOW  
&  
REPAINT**

**REFLOW  
&  
REPAINT**

**DO THIS ALOT = SLOW**

**ANY  
DOM  
UPDATE**

**MOVE /  
ANIMATE  
DOM**

**HIDE DOM  
NODE**

**STYLE  
CHANGES**

**MANAGEDOMSTATE**

**VIRTUALDOM**

**REDUCE NUMBER OF CHANGES  
TO BE APPLIED**

**SMART DIFF'ING OF DOM**

**REDUCE NUMBER OF  
REFLOWS / REPAINTS**

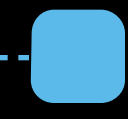
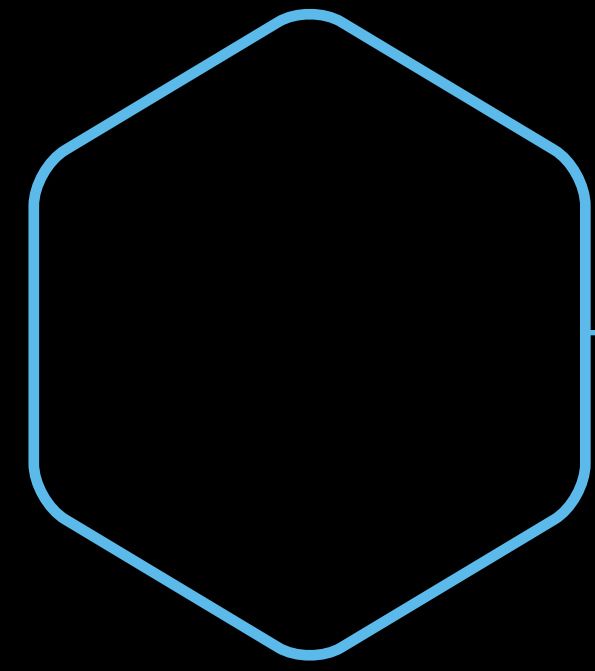
**BATCHED DOM UPDATES**

# REACT BASICS

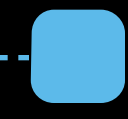
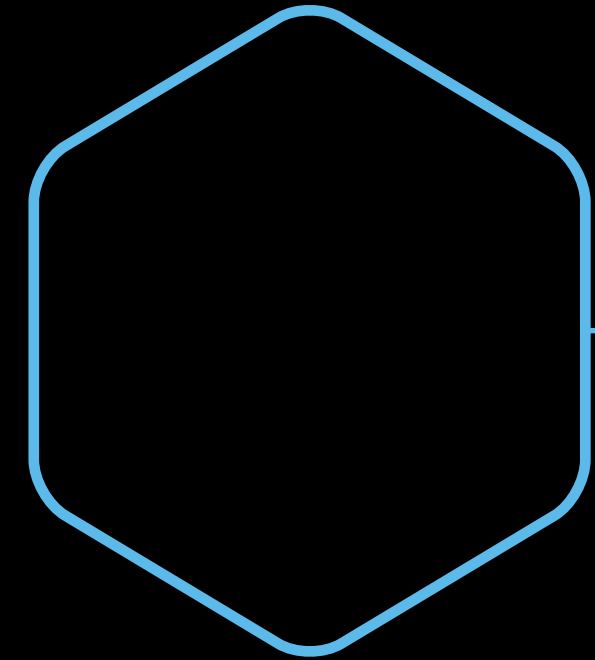




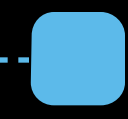
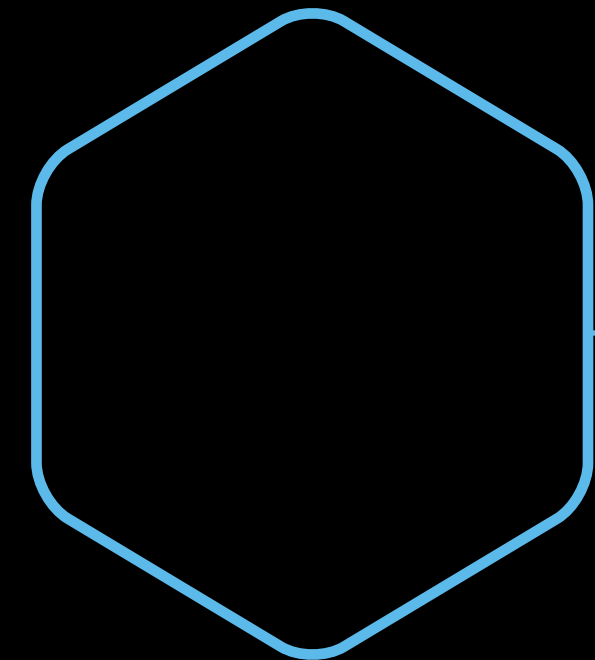
# COMPONENTS



**BASIC BUILDING BLOCKS**

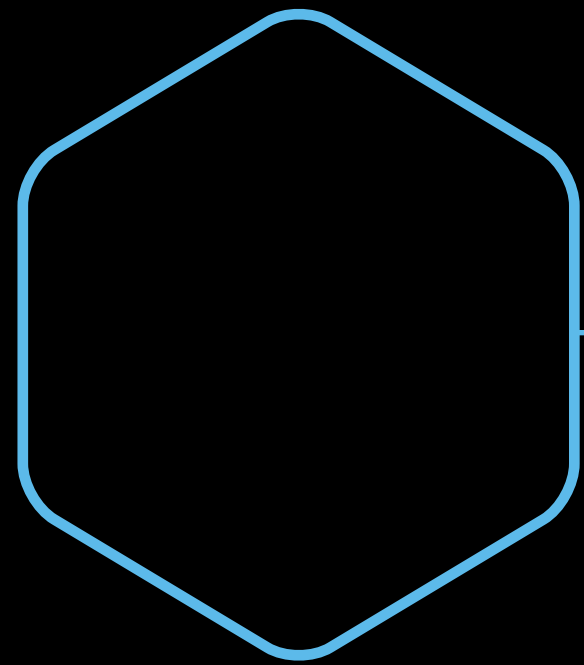


**JSX**

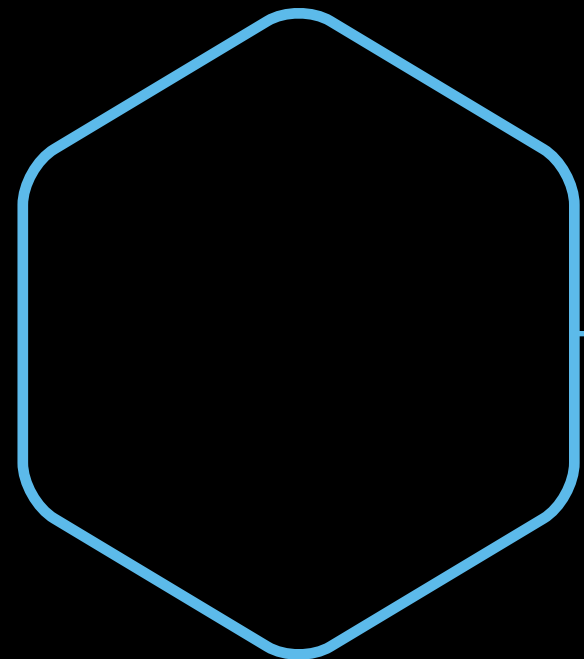


**JAVASCRIPT XML SYNTAX**

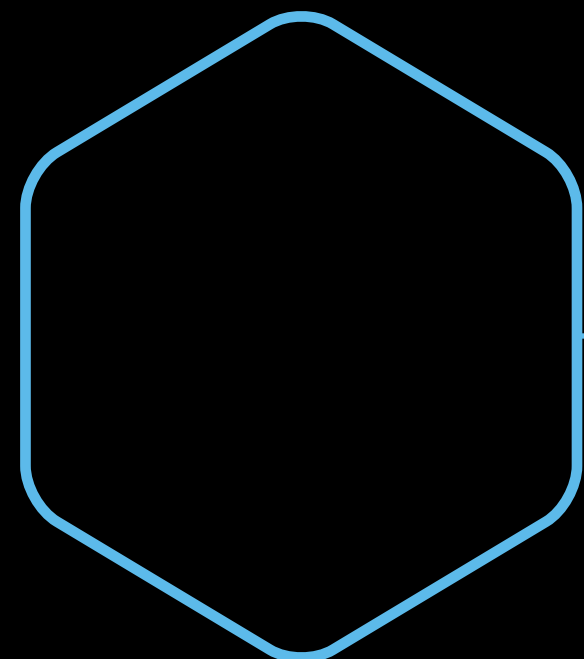
```
<script type="text/jsx">
/** @jsx React.DOM */
  React.renderComponent(
    <h1>Hello, world!</h1>,
    document.getElementById('myDiv')
  );
</script>
```



- **HTML-LIKE**



- **JSX DIFFERS FROM HTML**

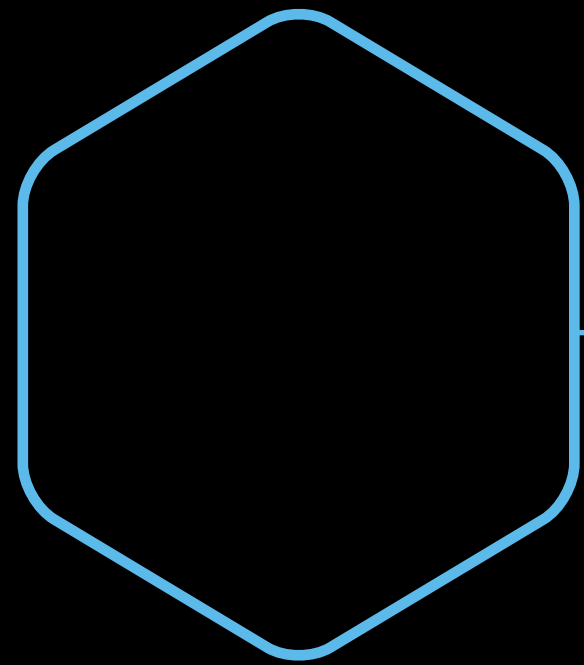


- **[HTTP://FACEBOOK.GITHUB.IO/REACT/DOCS/JSX-GOTCHAS.HTML](http://facebook.github.io/react/docs/jsx-gotchas.html)**

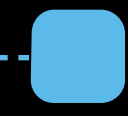
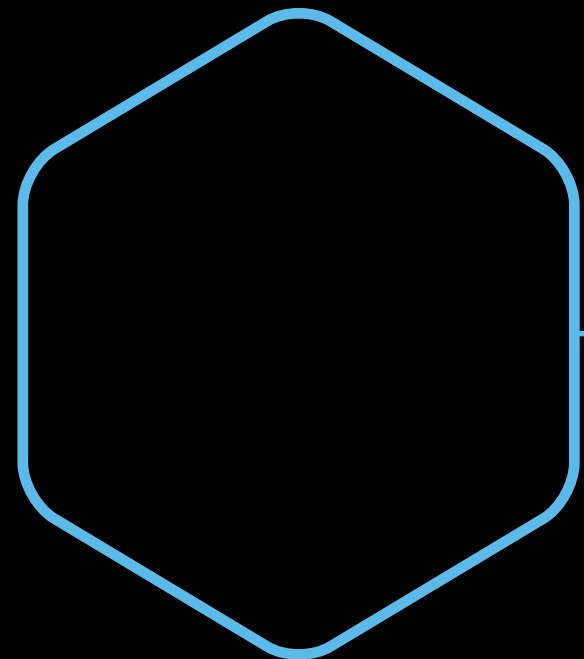


**WHY WRITE IN  
HTML-LIKE WHEN  
YOU CAN JUST  
WRITE CODE?**

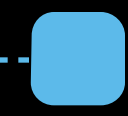
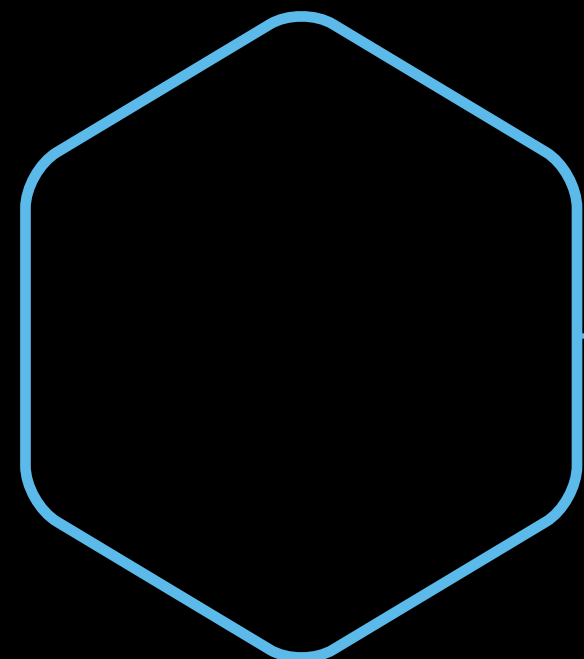
```
React.renderComponent(  
  React.DOM.h1(null, 'Hello, world!'),  
  document.getElementById('myDiv')  
);
```




**REACT.RENDERCOMPONENT**



**1: COMPONENT TO RENDER**



**2: WHERE TO MOUNT**

A collection of white-outlined geometric shapes, including rectangles and a triangle, arranged in a layered, overlapping fashion on the left side of the slide.

```
React.renderComponent(  
  component, whereToAttachToDOM  
);
```





# CUSTOM COMPONENTS: createClass

```
var MyComponent =  
  React.createClass({  
    render: function(){  
      return (<h1>Hello, world!</h1>);  
    }  
  });
```

```
// use component  
React.renderComponent(  
  <MyComponent/>,  
  document.getElementById('myDiv')  
);
```



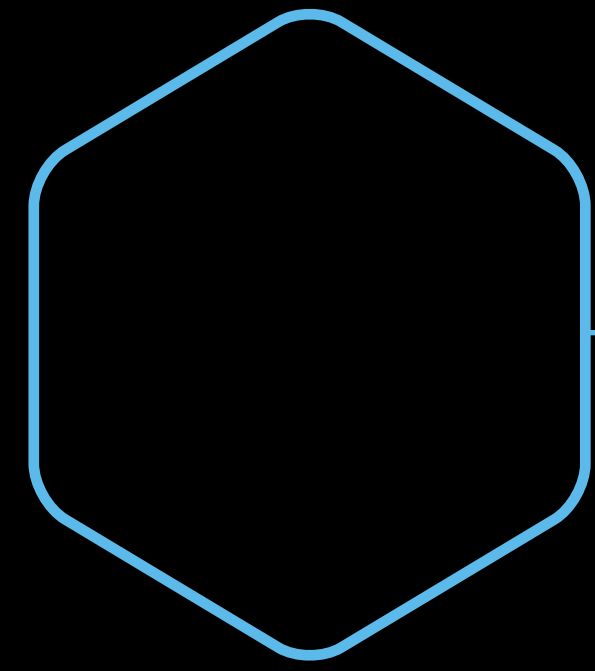
# DYNAMIC COMPONENTS: attributes

```
var MyComponent = React.createClass({  
  render: function(){  
    return (<h1>Hello, {this.props.name}</h1>);  
  }  
});
```

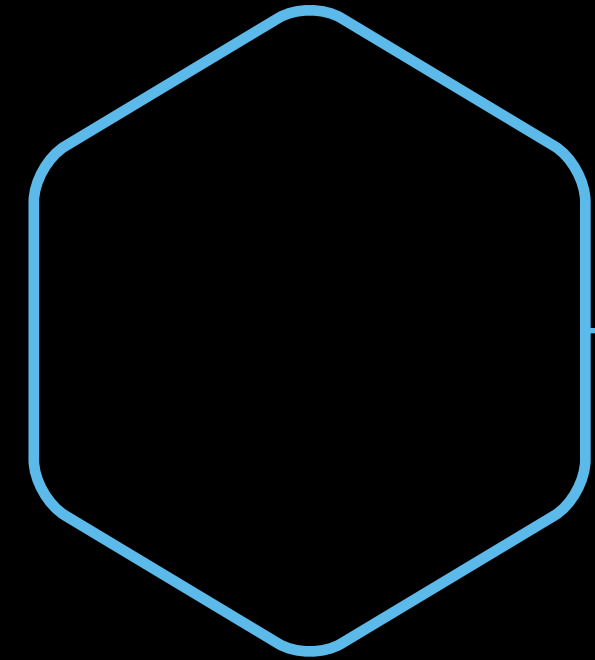
```
React.renderComponent(<MyComponent  
  name="Pratik" />,  
  document.getElementById('myDiv'));
```



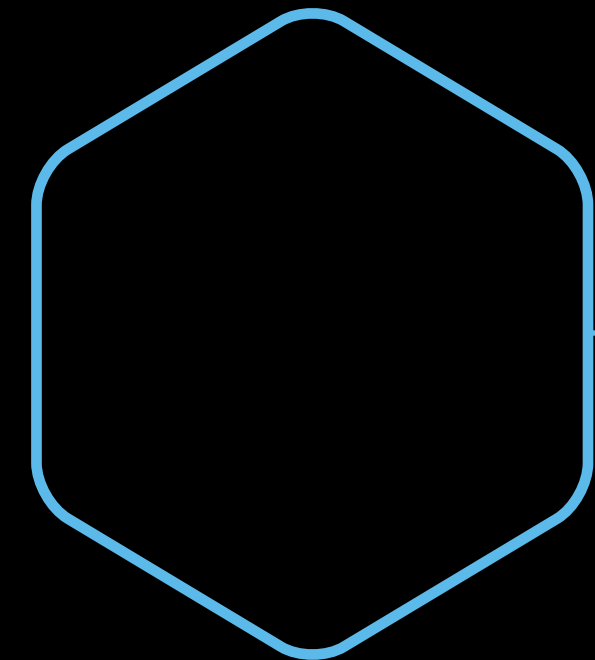
# COMPONENT SPECIFICATIONS



■ **getInitialState**



■ **getDefaultProps**



■ **mixins**

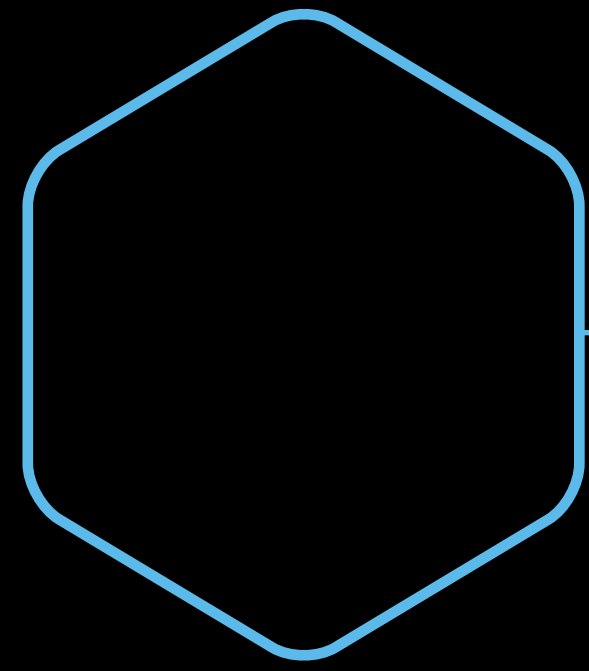


**MIXIN: sharing  
common lifecycle  
with cross-cutting  
concerns**

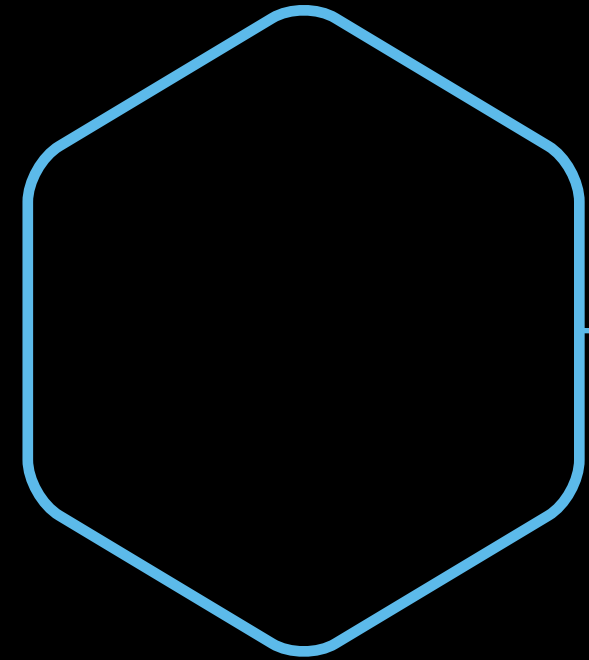


```
var LogMixin = {  
  componentDidMount: function () {  
    console.log("componentDidMount called!");  }  
};
```

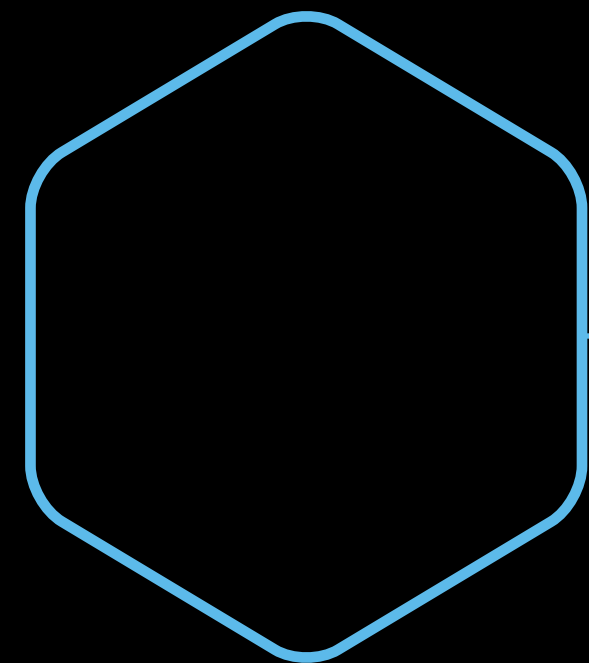
```
var ComponentTwo = React.createClass({  
  mixins: [LogMixin],  
  ..
```



- **componentWillRender**



- **componentDidMount**



- **componentWillUnmount**

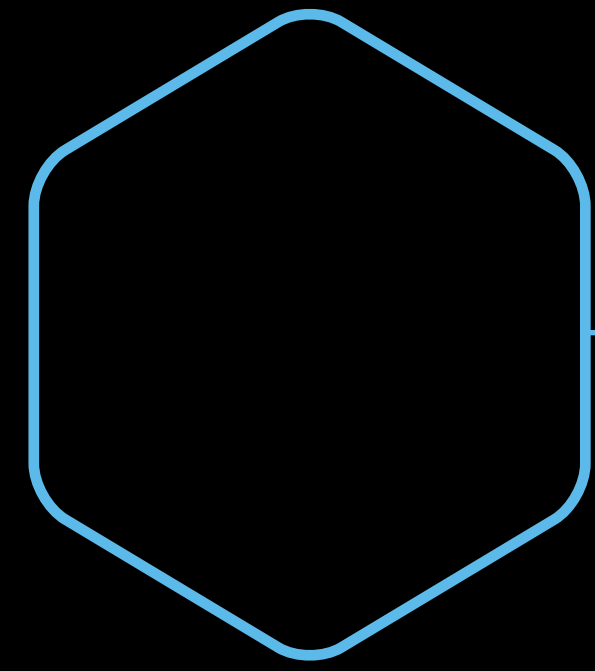


**COMPONENTS  
HAVE STATE**

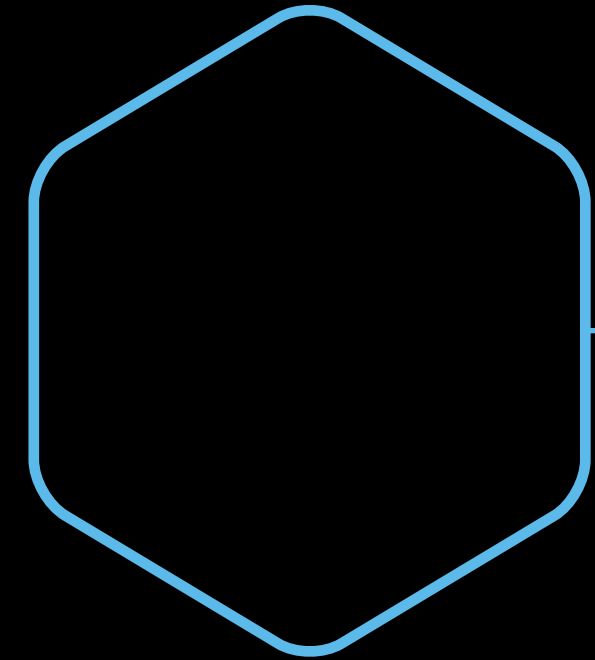
```
var MyComponent = React.createClass({
  getInitialState: function(){
    return {      count: 5      }
  },
  render: function(){
    return (
      <h1>{this.state.count}</h1>
    )
  }
});
```



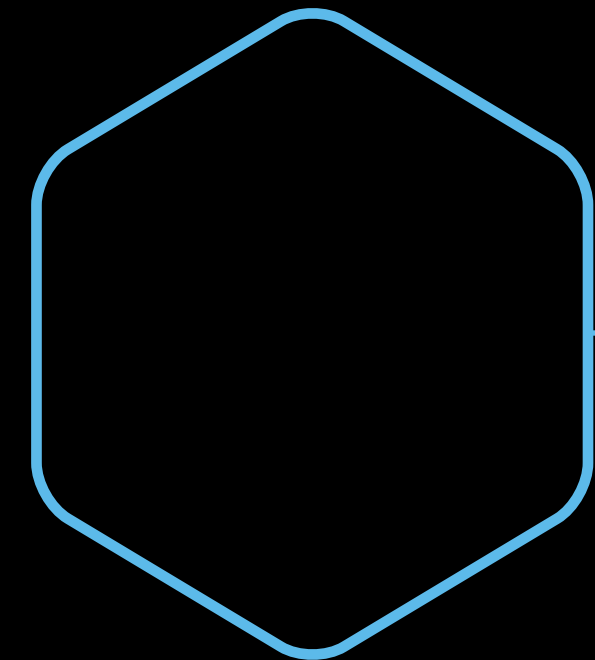
**EVENTS ARE  
WRAPPED!**



- **SyntheticEvent**



- **cross-browser wrapper**



- **wrap native event**

```
var Counter = React.createClass({
  incrementCount: function(){
    this.setState({
      count: this.state.count + 1
    });
  },
  getInitialState: function(){
    return { count: 0 }
  },
});
```

```
render: function(){
  return (
    <div class="my-component">
      <h1>Count: {this.state.count}</h1>
      <button type="button"
onClick={this.incrementCount}>
        Increment</button></div>  ); }
});
React.renderComponent(<Counter/>,
document.getElementById('mount-point'));
```

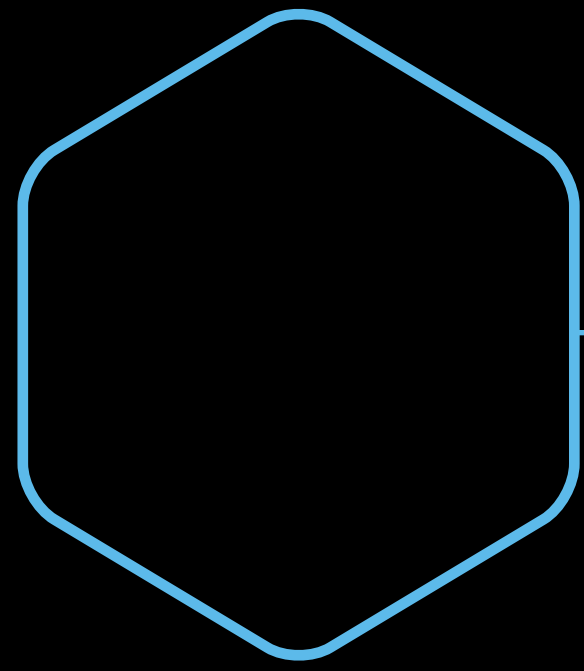


Changing the state  
causes a refresh:  
`this.setState(...)`

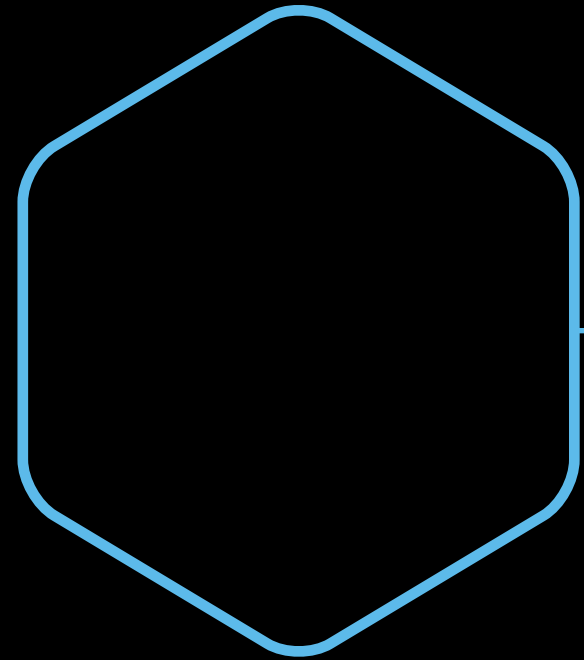




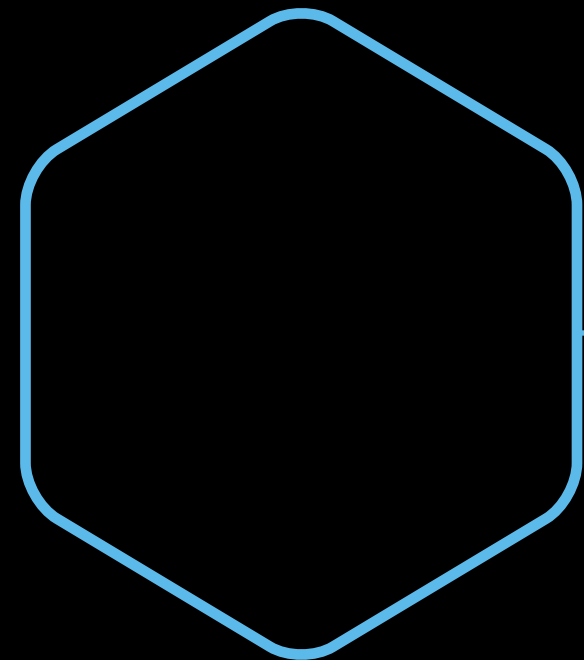
# TWITTER STREAM EXAMPLE



■ **Browserify**



■ **Socket.io**



■ **Combo browser/server  
render**

```
module.exports = TweetsApp = React.createClass({
  render: function(){
    return (
      <div className="tweets-app">
        <Tweets tweets={this.state.tweets} />
        <Loader paging={this.state.paging}/>
        <NotificationBar count={this.state.count}
onShowNewTweets={this.showNewTweets}/>
      </div>
    )
  });
```

```
/** @jsx React.DOM */  
var React = require('react');  
var Tweet = require('./Tweet.react.js');  
module.exports = Tweets = React.createClass({  
  render: function(){  
    var content =  
this.props.tweets.map(function(tweet){  
  return (<Tweet key={tweet.twid} tweet={tweet} />);  
});  
    return (<ul className="tweets">{content}</ul>);  
  });  
});
```

```
module.exports = Tweet = React.createClass({
  render: function(){
    var tweet = this.props.tweet;
    return (
      <li className={"tweet" + (tweet.active ? ' active' : "")}>
        <img src={tweet.avatar} className="avatar"/>
        <blockquote>
          <cite>
            <a href={"http://www.twitter.com/" +
tweet.screenname}>{tweet.author}</a>
```

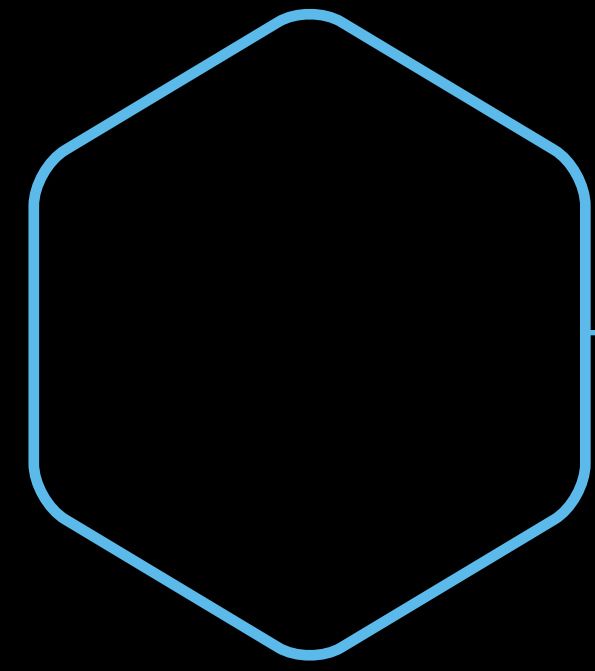


ISOMORPHIC

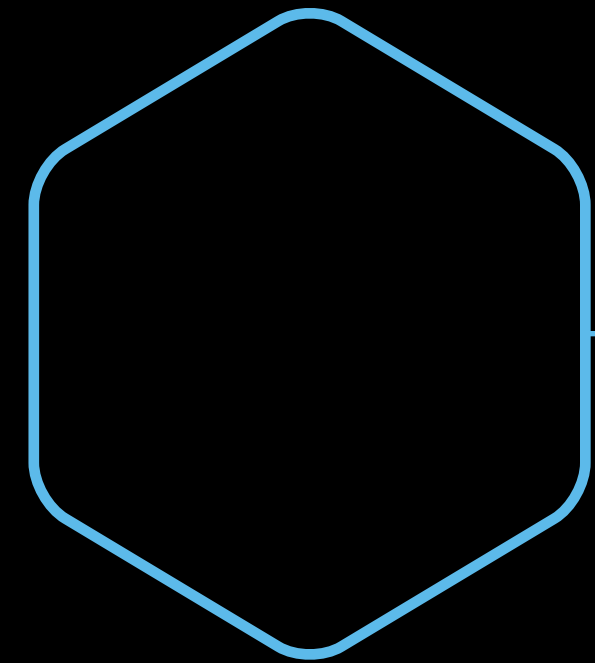
```
var JSX = require('node-jsx').install(),  
    React = require('react'),  
    TweetsApp = require('./components/  
TweetsApp.react'),  
    Tweet = require('./models/Tweet');
```



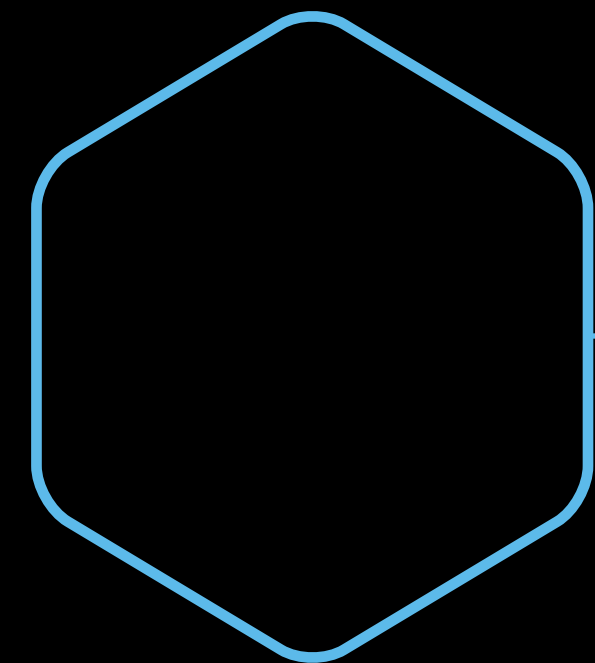
```
var markup = React.renderComponentToString(  
  TweetsApp({ tweets: tweets })  
);  
// Render our 'home' template  
res.render('home', {  
  markup: markup, // Pass rendered react markup  
  state: JSON.stringify(tweets)  
});
```



**Generate initial page on server**



**New data rendered in browser**



**“infinite scroll” on browser**



# BROWSER

```
// TweetApps.react.js (main component)
showNewTweets: function(){
  var updated = this.state.tweets;
  updated.forEach(function(tweet){
    tweet.active = true;
  });
  // Forces render!
  this.setState({tweets: updated, count: 0});
},
```

`setState(...)`  
invokes render for  
component and all  
sub components!



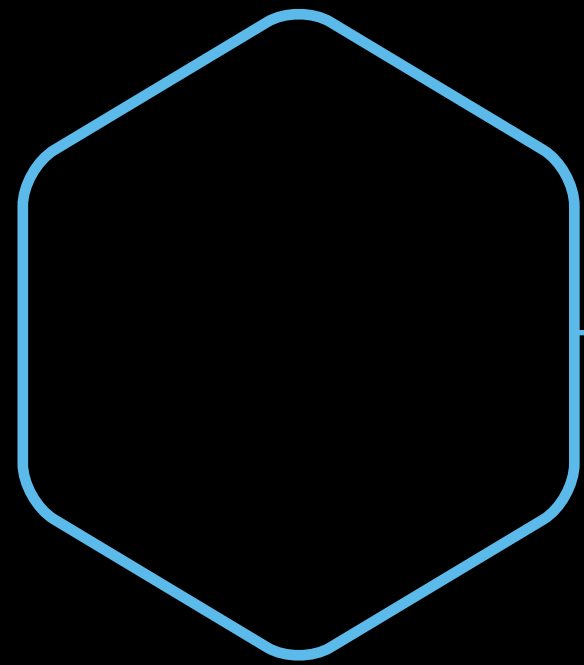
**WON'T  
THAT BE  
SUPER  
SLOW?**



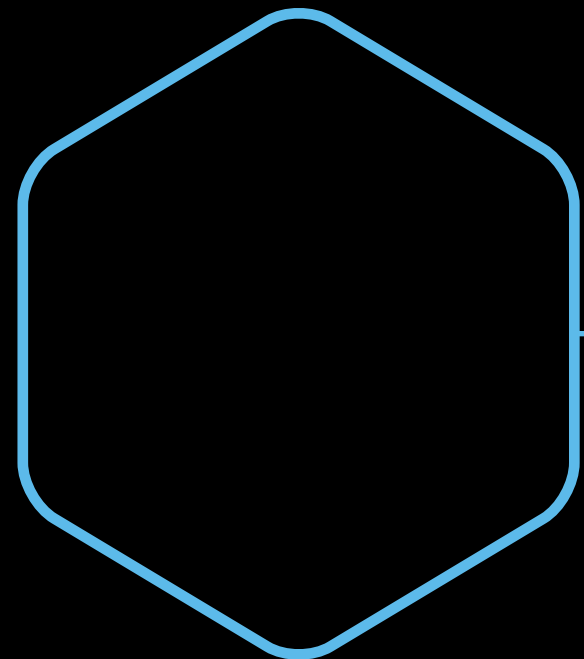
**NO,  
NOT IN  
REACT**

**REMEMBER THE VIRTUAL DOM  
AND THE COOL STUFF IT DOES!**

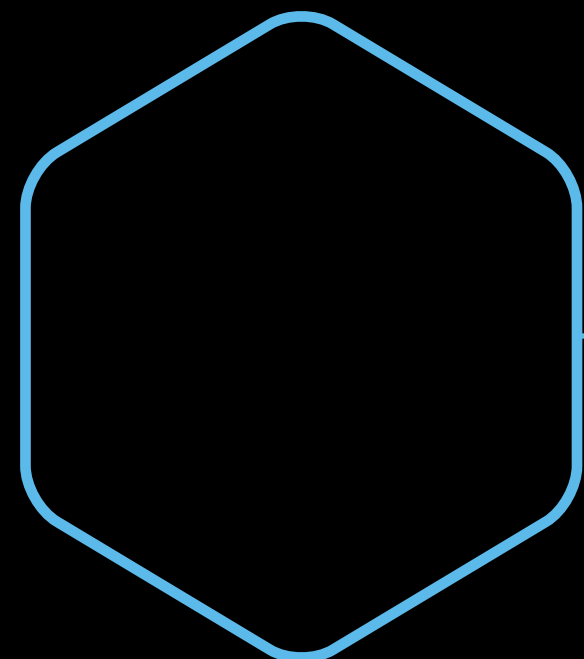
[@prpatel](#)



**VIRTUAL DOM**



**SMART DIFF'ING OF DOM**



**BATCHED DOM UPDATES**





**SERVER**

```
var markup = React.renderComponentToString(  
  TweetsApp({ tweets: tweets })  
);  
// Render our 'home' template  
res.render('home', {  
  markup: markup, // Pass rendered react markup  
  state: JSON.stringify(tweets)  
});
```



**THANK YOU**

References:

<http://scotch.io/tutorials/javascript/learning-react-getting-started-and-concepts>

<http://facebook.github.io/react/index.html>

@prpatel