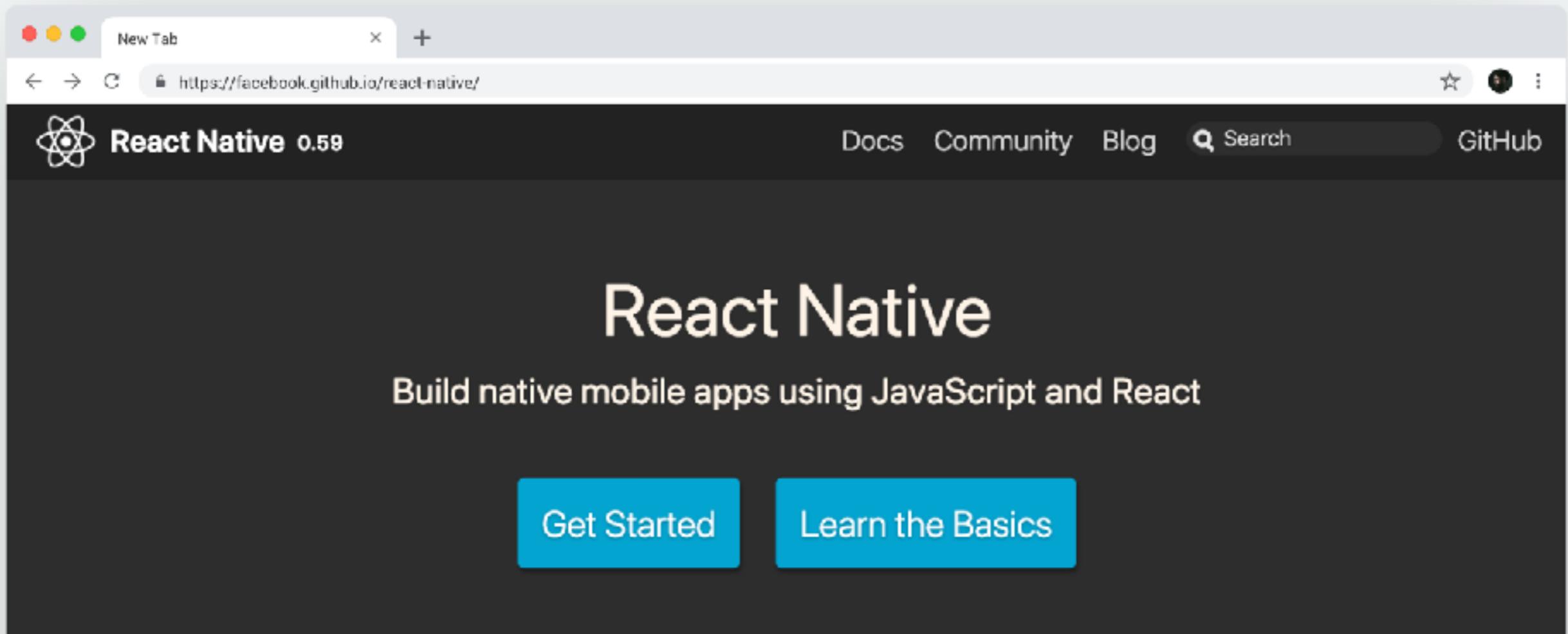


React Native 101

github.com/alexisleon

React Native?

A **JavaScript framework** used to **build** mobile **apps**



React Native

Next gen of ReactJS.

Developed by **Facebook** and **Instagram**.
Released on Github in 2013.

Main idea:

Engineers won't have to build the same app for
iOS and for Android from scratch
- reusing code **across operating systems.**

React Native?

Pros:

The community

Faster development

Closer teams

Cross-platform building

Cons:

It's still improving.

Why that much faith?

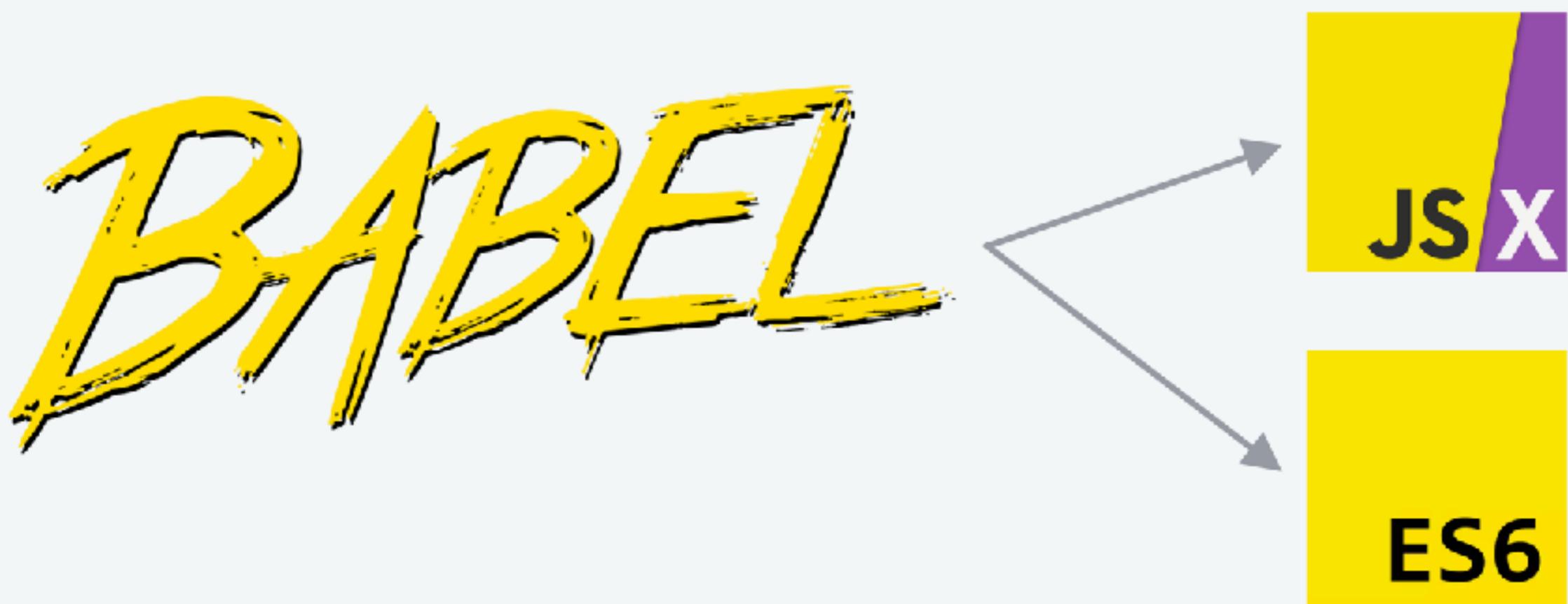
1. Saves companies a lot of resources
2. The only threat is Facebook cutting off the project
- 3 Lots of companies and Facebook themselves
heavily rely on it





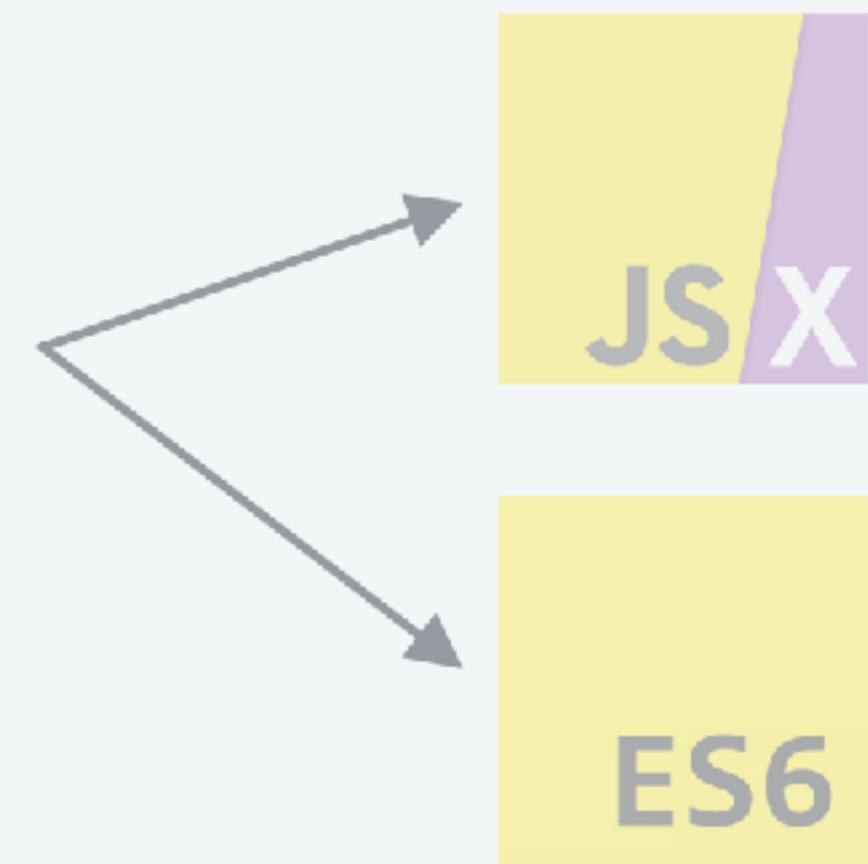
```
<script type="text/javascript">
    //JavaScript goes here
</script>
```

Modern JavaScript



BABEL

Preprocessor



```
[1,2,3].map(n => n + 1);
```

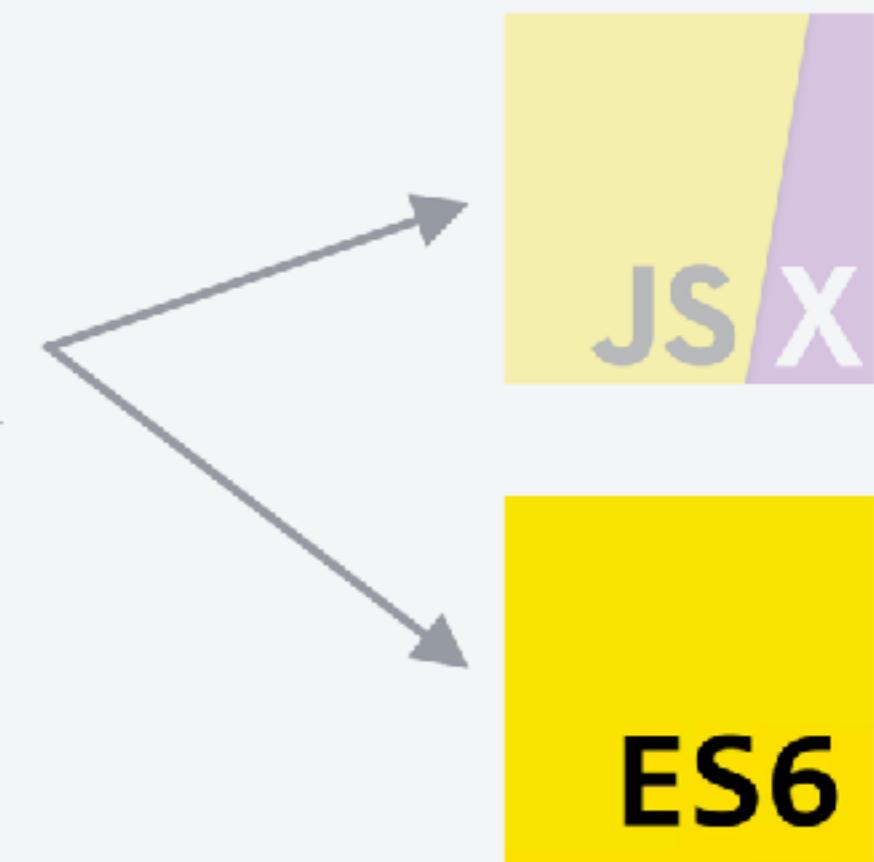


BABEL



```
[1, 2, 3].map(function (n) {  
    return n + 1;  
});
```

BABEL



ES6

(ECMAScript 6)

New JS features
Better code

```
const age = 32  
let day = 'Monday'
```

```
const dob = (yrs) => 2019 - yrs  
dob(age) // 1987
```

```
const fruit = ['apple', 'orange']  
const food = [...fruit, 'potato']
```

ES6



```
const a = 1
// not allowed!
// a = 2
```

```
let b = 'foo'
b = 'bar' // this is ok
if (true) {
  const a = 3
}
```



```
var a = 1;
// not allowed!
// a = 2
```

```
var b = 'foo';
b = 'bar'; // this is ok
if (true) {
  var _a = 3;
}
```

ES6



```
const people = ['eva', 'john', 'mark']
const newPeople = [...people]
const lotsOfPeople = [...people, 'trent', 'lolita', 'james']

const [one, two, ...rest] = people
```

BABEL



```
var people = ['eva', 'john', 'mark'];
var newPeople = [].concat(people);
var lotsOfPeople = [].concat(people, ['trent', 'lolita', 'james']);

var one = people[0],
    two = people[1],
    rest = people.slice(2);
```

ES6



```
const foo = () => 'bar'

const baz = (x) => {
  return x + 1
}

this.items.map(x => this.doSomethingWith(x))
```

BABEL



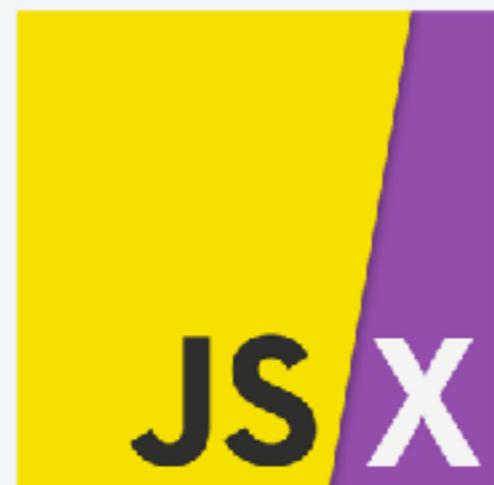
```
var _this = this;

var foo = function foo() {
  return 'bar';
};

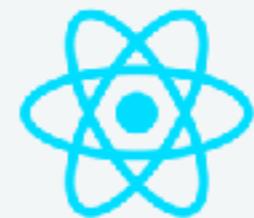
var baz = function baz(x) {
  return x + 1;
};

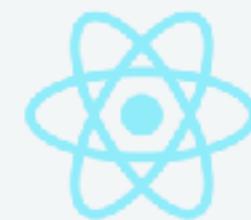
this.items.map(function (x) {
  return _this.doSomethingWith(x);
});
```

ES6 → ES7 → ES8



An extension to JavaScript that
you will use to build your UI
interfaces.

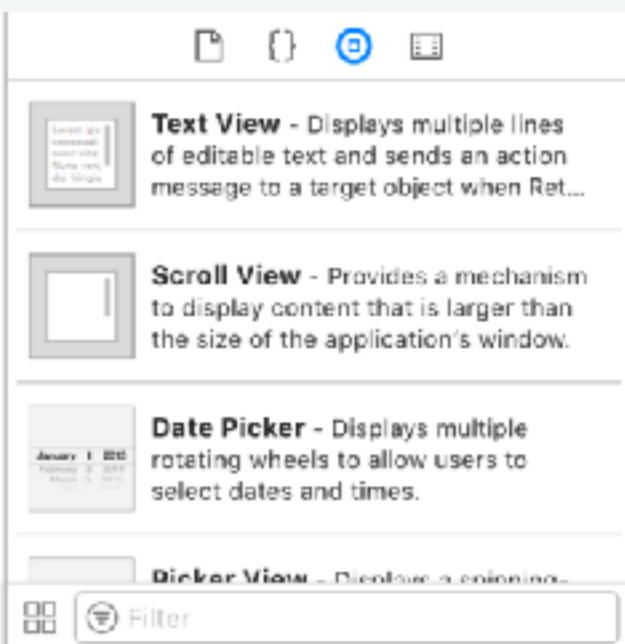
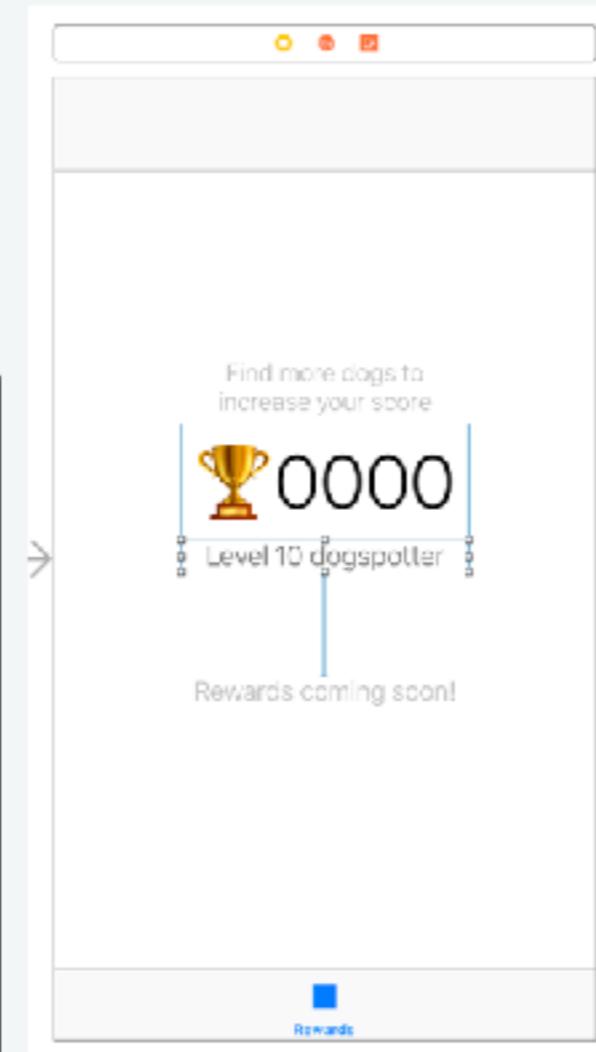


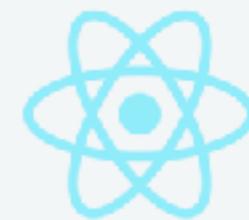




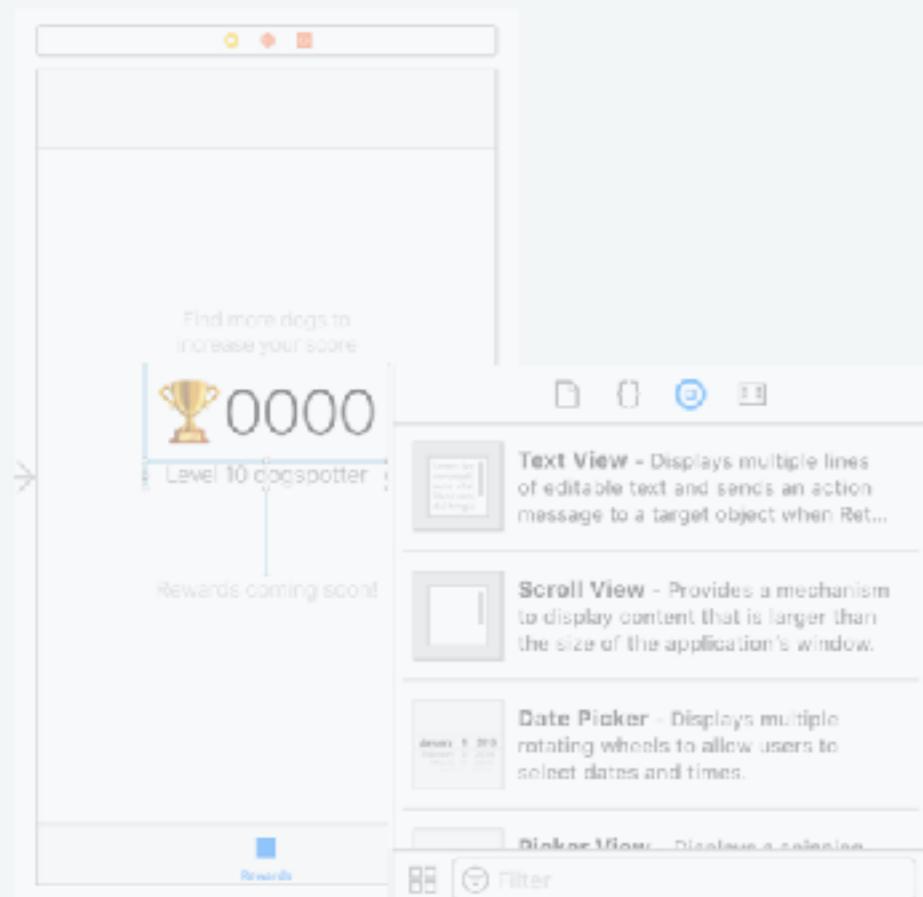
```
let textLayer = CATextLayer()
textLayer.backgroundColor = color.cgColor
textLayer.foregroundColor = UIColor.white.cgColor
textLayer.frame = frame
textLayer.alignmentMode = kCAAlignmentLeft
textLayer.isWrapped = true
let font = CTFontCreateWithName("System" as CFString, 18.0, nil)
textLayer.font = font
textLayer.fontSize = 18.0
textLayer.contentsScale = UIScreen.main.scale
textLayer.string = label

self.view.layer.addSublayer(textLayer)
```



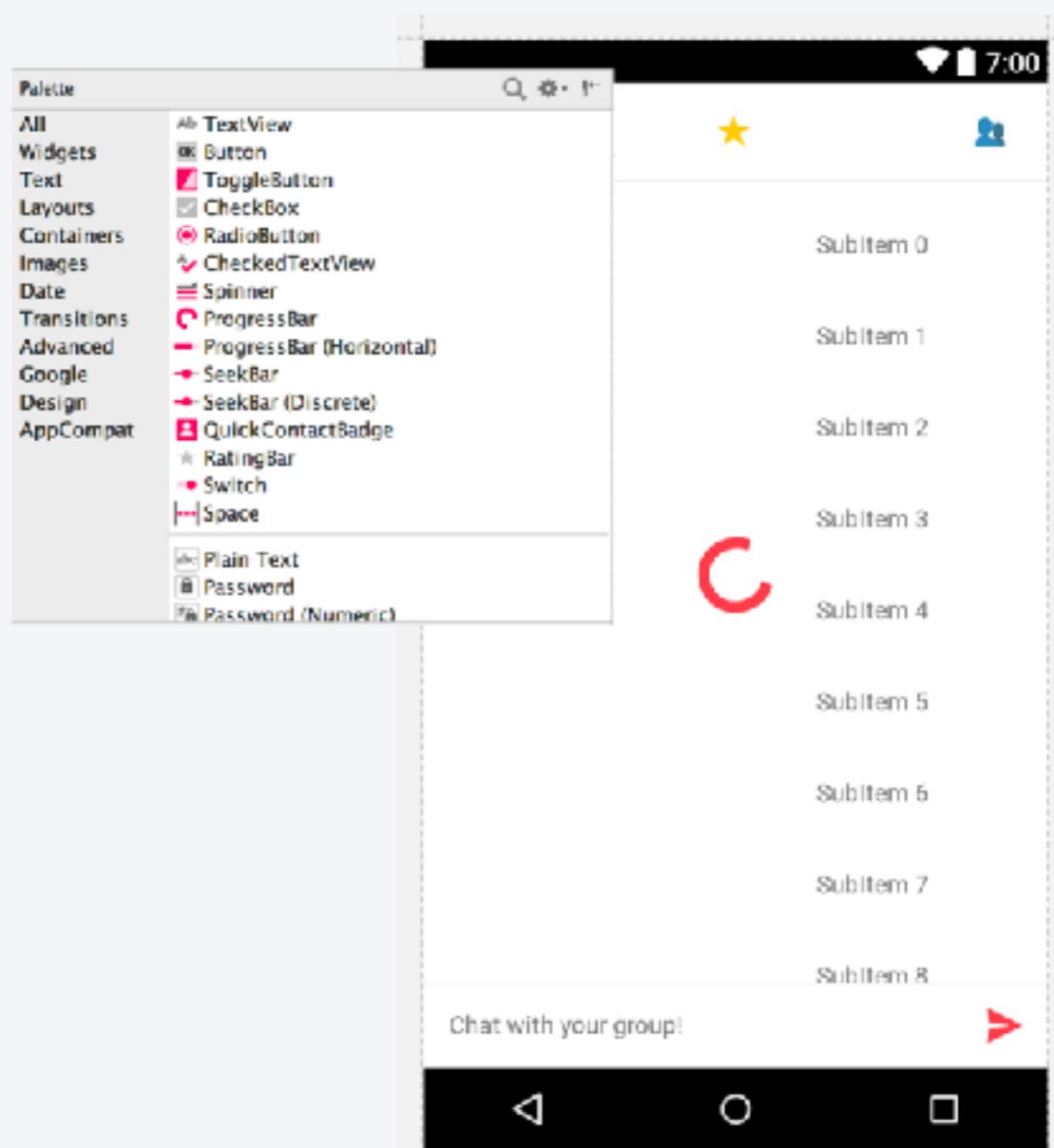


```
let textLayer = CATextLayer()  
...  
self.view.layer.addSublayer(textLayer)
```



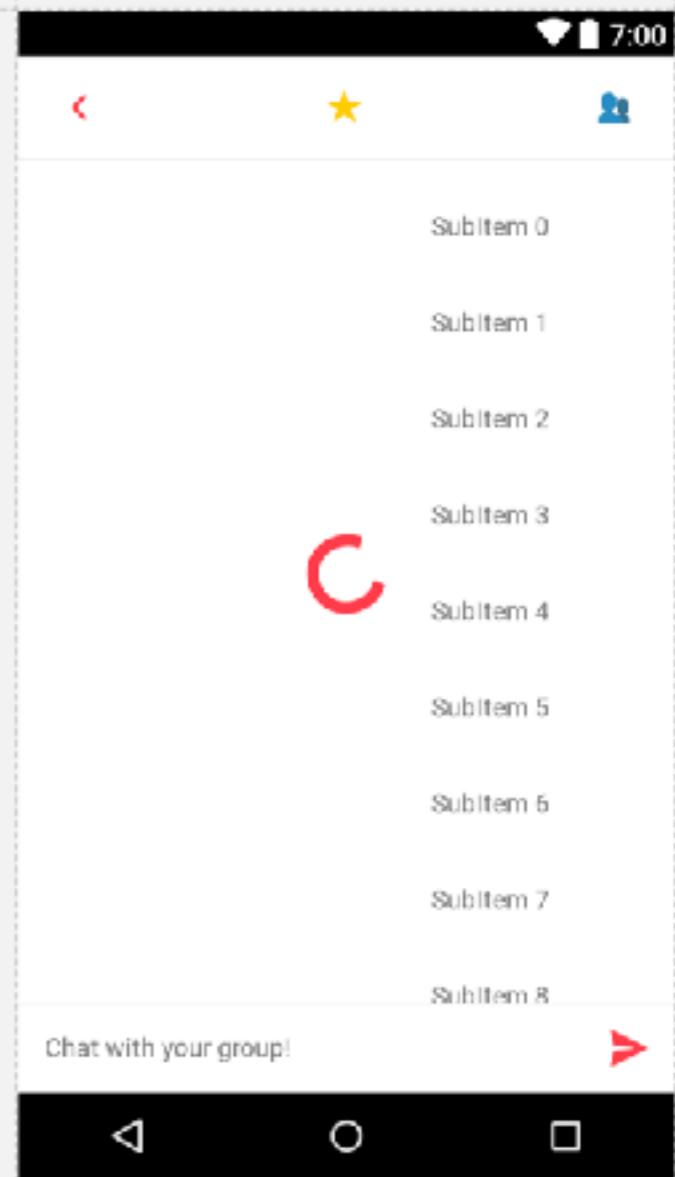
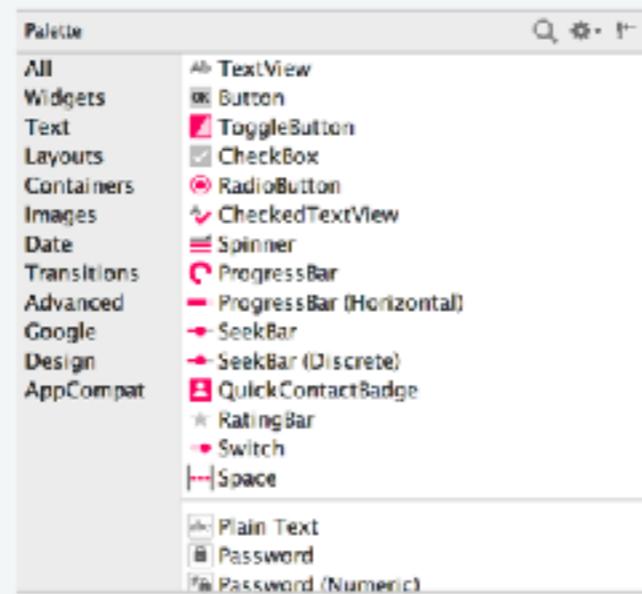


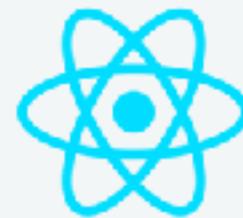
```
LinearLayout myLayout = findViewById(R.id.main);  
  
Button myButton = new Button(this);  
myButton.setLayoutParams(new  
    LinearLayout.LayoutParams(  
        LinearLayout.LayoutParams.MATCH_PARENT,  
        LinearLayout.LayoutParams.MATCH_PARENT));  
  
myLayout.addView(myButton);
```



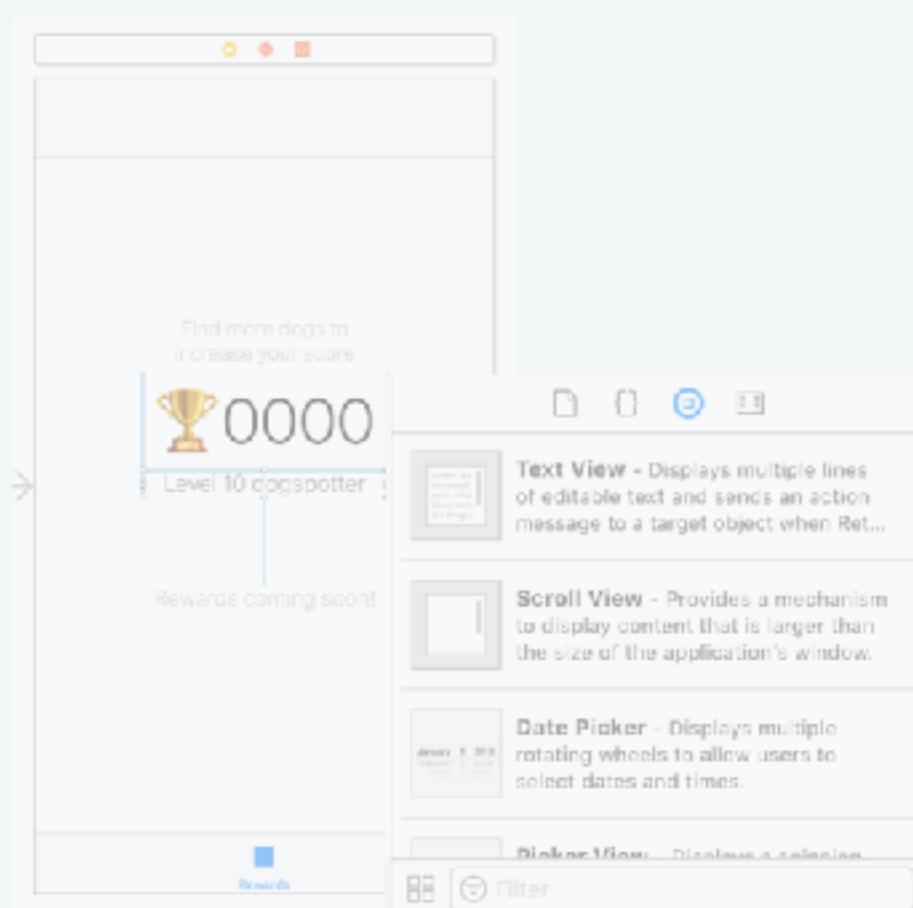


```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@android:color/white"  
    android:gravity="center"  
    android:minHeight="48dp"  
    android:orientation="horizontal">  
  
    <EditText  
        android:id="@+id/chat_message_text"  
        android:layout_width="0dp"  
        android:layout_height="wrap_content"  
        android:layout_weight="1"  
        android:background="@android:color/transparent"  
        android:ems="10"  
        android:enabled="false"  
        android:hint="Chat with your group!"  
        android:inputType="textMultiLine|textCapSentences"  
        android:maxLines="5"  
        android:paddingEnd="5dp"  
        android:paddingLeft="15dp"  
        android:paddingRight="5dp"  
        android:paddingStart="15dp"  
        android:textAppearance="?android:attr/textAppearanceSmall" />  
  
    <!--<ImageButton-->  
    <!--> android:id="@+id/chat_message_attach"-->  
    <!--> android:layout_width="wrap_content"-->  
    <!--> android:layout_height="wrap_content"-->  
    <!--> android:background="?attr/selectableItemBackgroundBorderless"-->  
    <!--> android:paddingBottom="10dp"-->  
    <!--> android:paddingLeft="5dp"-->  
    <!--> android:paddingRight="5dp"-->  
    <!--> android:paddingTop="10dp"-->  
    <!--> android:src="@drawable/ic_attach" /-->  
  
</LinearLayout>
```

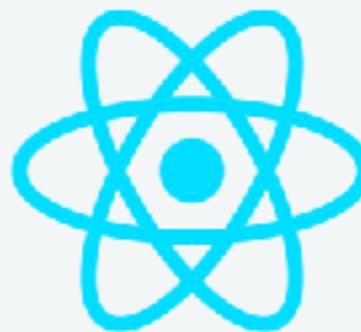




```
let textLayer = CATextLayer()  
...  
self.view.layer.addSublayer(textLayer)
```

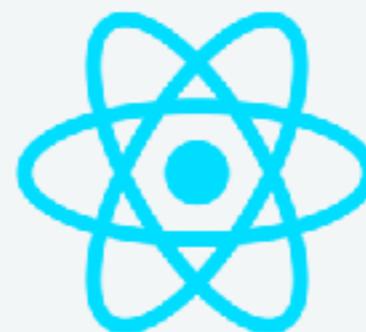


```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@android:color/white"  
    android:gravity="center"  
    android:minHeight="48dp"  
    android:orientation="horizontal">   
  
<EditText  
    android:id="@+id/chat_message_text"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:background="@android:color/  
    android:ems="18"  
    android:enabled="false"  
    android:hint="Chat with your group!"  
    android:inputType="textMultiLine|te  
    android:maxLines="5"  
    android:paddingEnd="5dp"  
    android:paddingLeft="15dp"  
    android:paddingRight="5dp"  
    android:paddingStart="15dp"  
    android:textAppearance="?android:at  
  
</EditText>  
<!--<ImageButton-->  
<!--android:id="@+id/chat_message_a  
<!--android:layout_width="wrap_content  
<!--android:layout_height="wrap_content  
<!--android:background="?attr/selec  
<!--android:paddingBottom="10dp"-->  
<!--android:paddingLeft="5dp"-->  
<!--android:paddingRight="5dp"-->  
<!--android:paddingTop="10dp"-->  
<!--android:src="drawable/ic_attach  
    Chat with your group!"/>   
</LinearLayout>
```



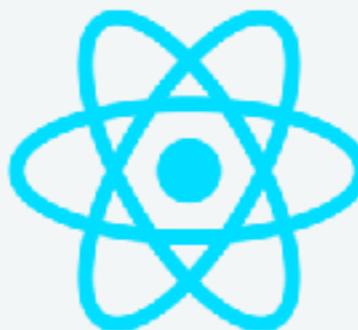
How do we build layouts in React Native?

Programmatically (in markup style) — this means no drag and drop interface builder.



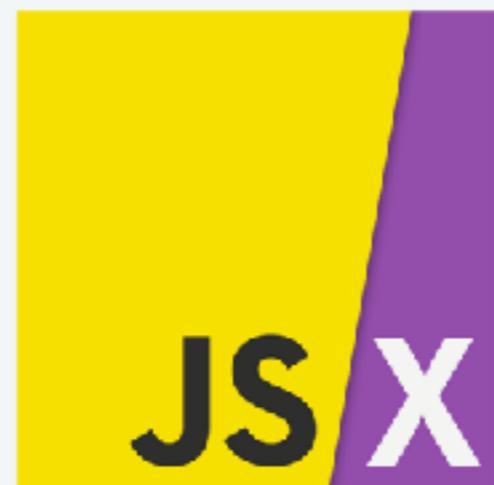
Originally, creating a View using React looked like this:

```
var ourView = React.createElement(View, null);
```



Originally, creating a View using React looked like this:

```
var ourView = React.createElement(View, null);
var ourNestedView = React.createElement(
  View,
  {
    foo: 'bar'},
  React.createElement(
    Text,
    null,
    '42'
  )
);
```



An extension to JavaScript that
you will use to build your UI
interfaces.

Without JSX

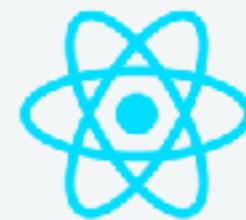
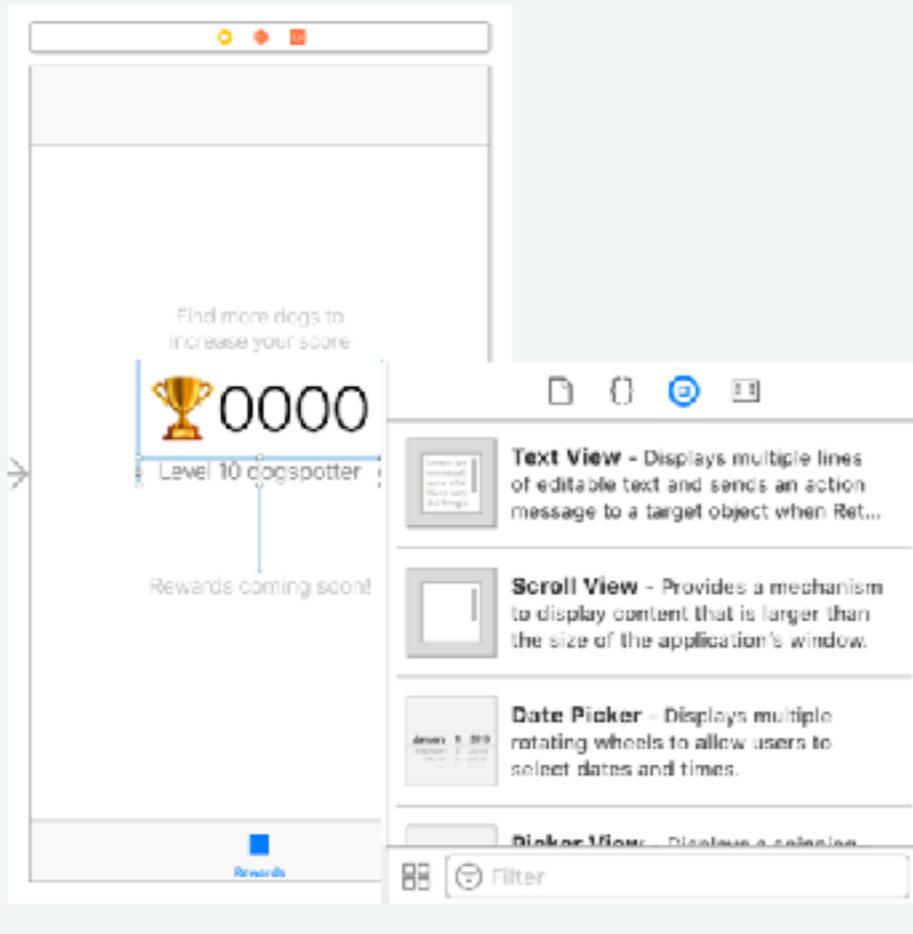
```
var ourNestedView = React.createElement(  
  View,  
  {  
    foo: 'bar'},  
  React.createElement(  
    Text,  
    null,  
    '42'  
  )  
);
```

With JSX

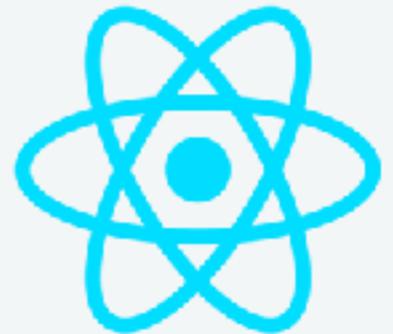
```
const ourNestedView = (  
  <View  
    foo='bar'>  
    <Text>42</Text>  
  </View>  
)
```



```
let textLayer = CATextLayer()  
...  
self.view.layer.addSublayer(textLayer)
```



```
const ourNestedView = (  
  <View  
    foo='bar'>  
    <Text>42</Text>  
  </View>  
)
```



```
const ourNestedView = (
  <View
    foo='bar'>
    <Text>42</Text>
  </View>
)
```

The screenshot shows the React Native documentation page with three examples:

- Image - deco**: Displays an image from a remote URL or a static asset. It includes a screenshot of a beach scene.
- Map View - deco**: Displays a native map with optional annotations and overlays. It includes a screenshot of a map of North America.
- Modal - deco**: Displays content in a view which overlays the entire app. It includes a screenshot of a modal overlay.

On the right side of the screenshot, there is a code editor window showing the following code snippet:

```
14 class Project extends Component {
15   render() {
16     return (
17       <View style={styles.container}>
18         <Text style={styles.welcome}>
19           Welcome to React Native!
20         </Text>
21         <Text style={styles.instructions}>
22           To get started, edit index.ios.js
23         </Text>
24         <Text style={styles.instructions}>
25           Press Cmd+R to reload, ('n')
26         </Text>
27       </View>
28     )
29   }
30 }
```

Setup

You need JavaScript in your terminal



Setup

You need a dependency manager



yarn

or



Setup



or



Install React Native + Deps

```
brew install node  
brew install watchman
```

```
npm install -g react-native-cli
```

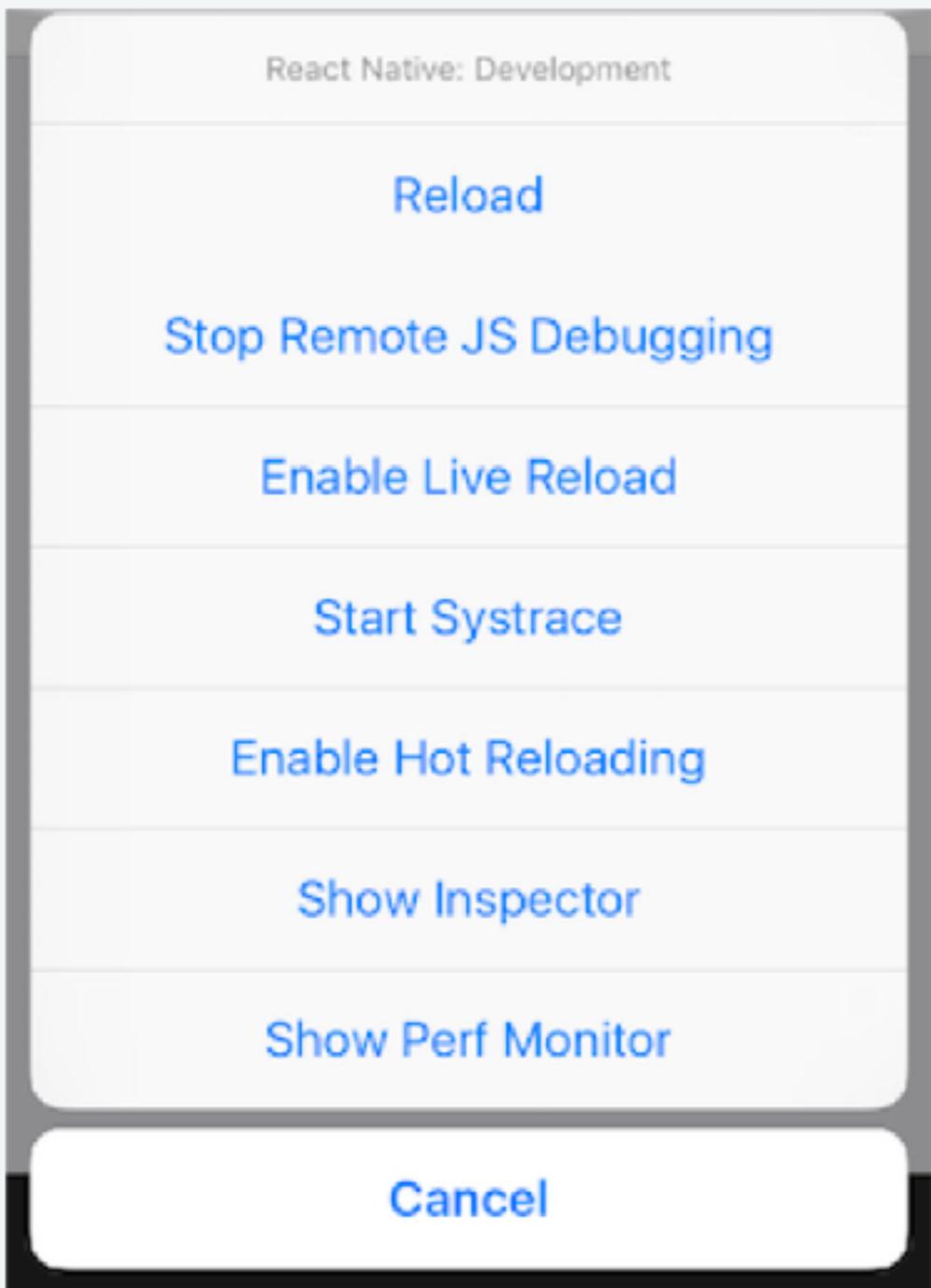
<https://facebook.github.io/react-native/docs/getting-started>

First RN App

react-native init AwesomeProject

Lets play with RN!

In-App Developer Menu



⌘M on Android

⌘D on iOS

Live Reload

Make a change, app reloads

Hot Reloading

Make a change, inject new versions of
your files to JS bundle

Rebuild for native changes

Images.xcassets or res/drawable

Native Modules (Obj-C, Swift, Java, Kotlin)

Errors and Warnings

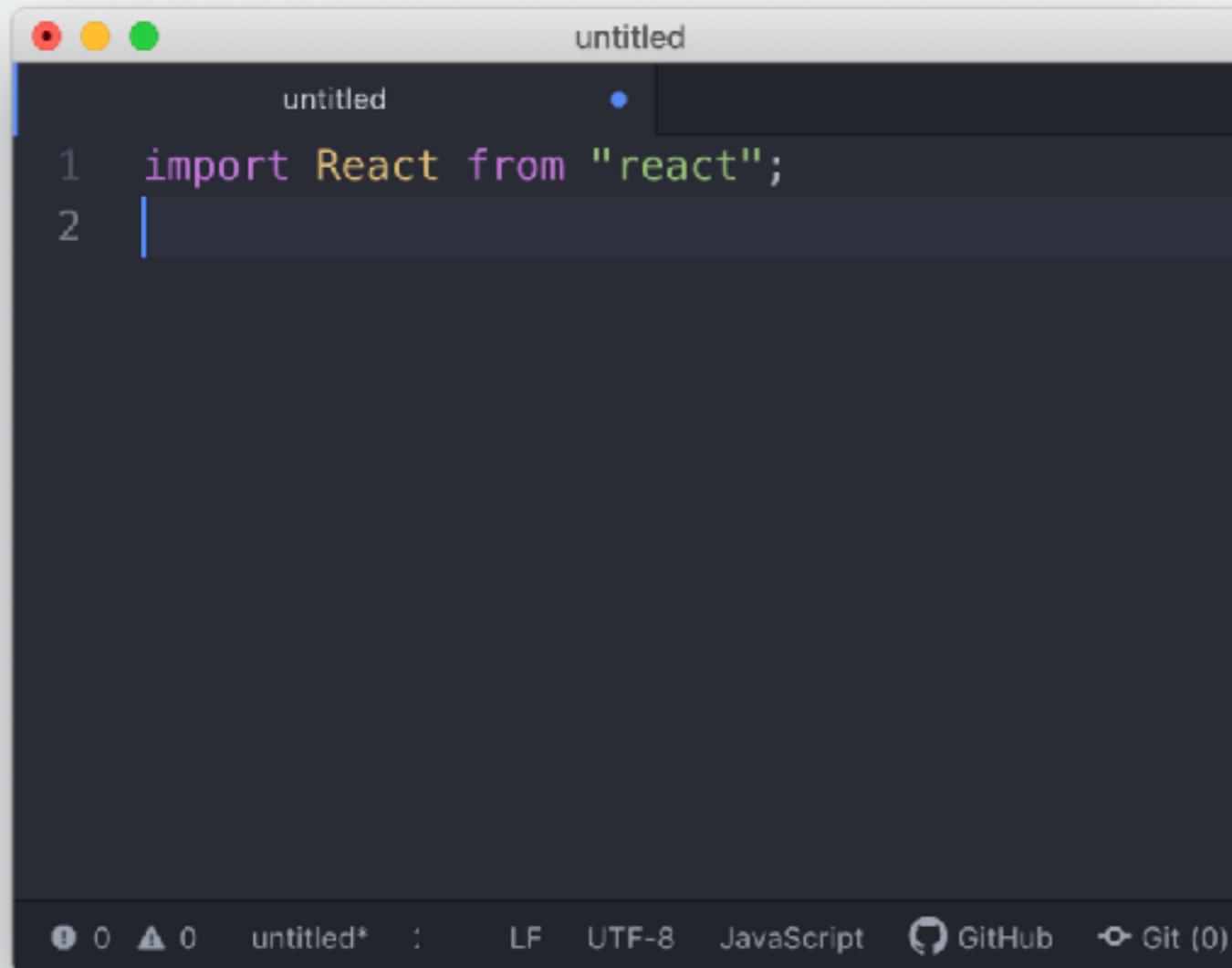
`console.error()`

`console.warn()`

`IS_TESTING`

RedBoxes and **YellowBoxes** are automatically disabled in **release** (production) builds.

First Component



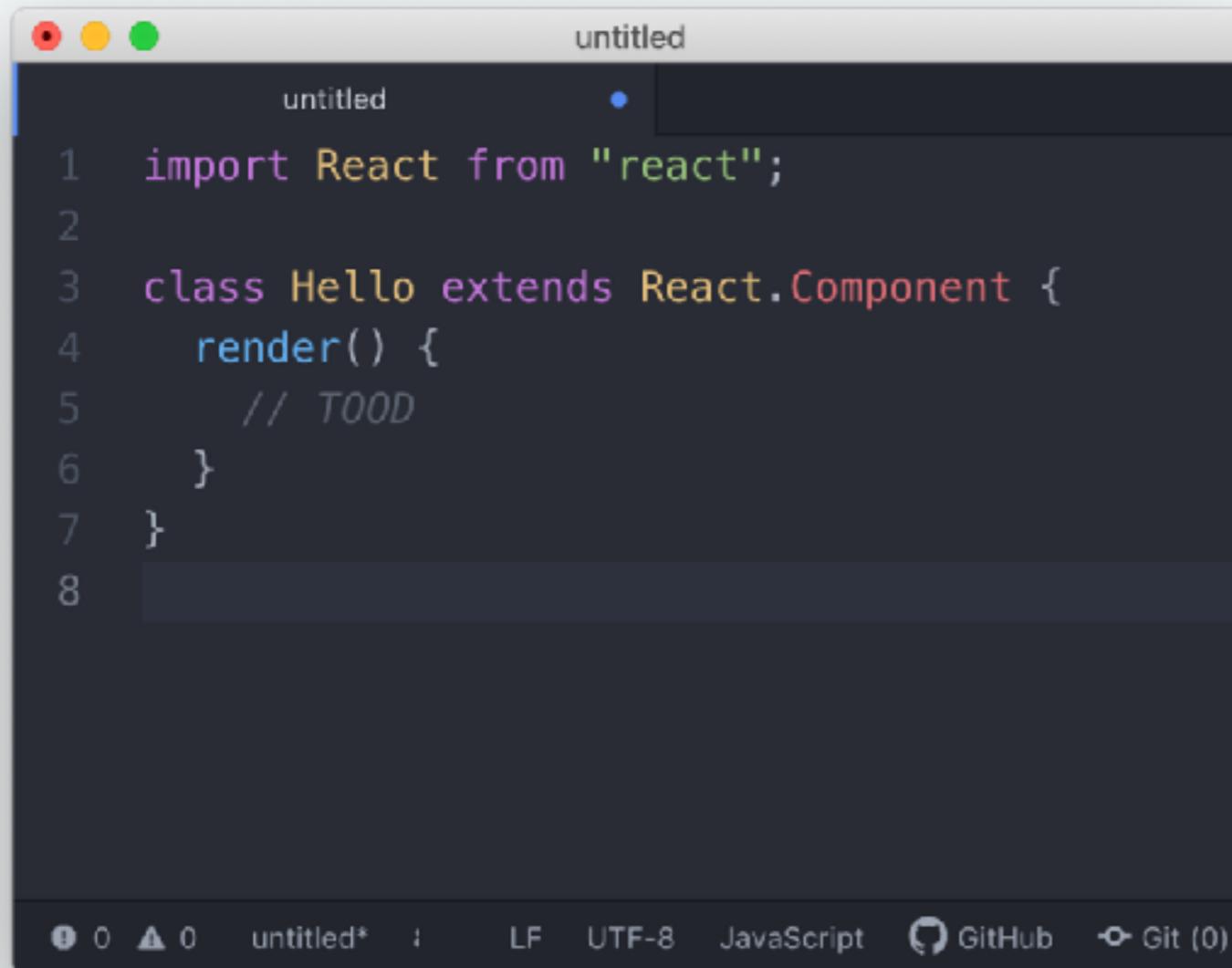
A screenshot of a code editor window titled "untitled". The code editor has a dark theme with light-colored syntax highlighting. The file is named "untitled" and contains the following code:

```
1 import React from "react";
2 |
```

The editor interface includes standard OS X-style window controls (red, yellow, green) at the top left. At the bottom, there are status icons for file count (0), alert count (0), and a GitHub integration icon. The bottom bar also shows the file name "untitled*", encoding "UTF-8", language "JavaScript", and a Git status showing 0 changes.

1. Import React

First Component



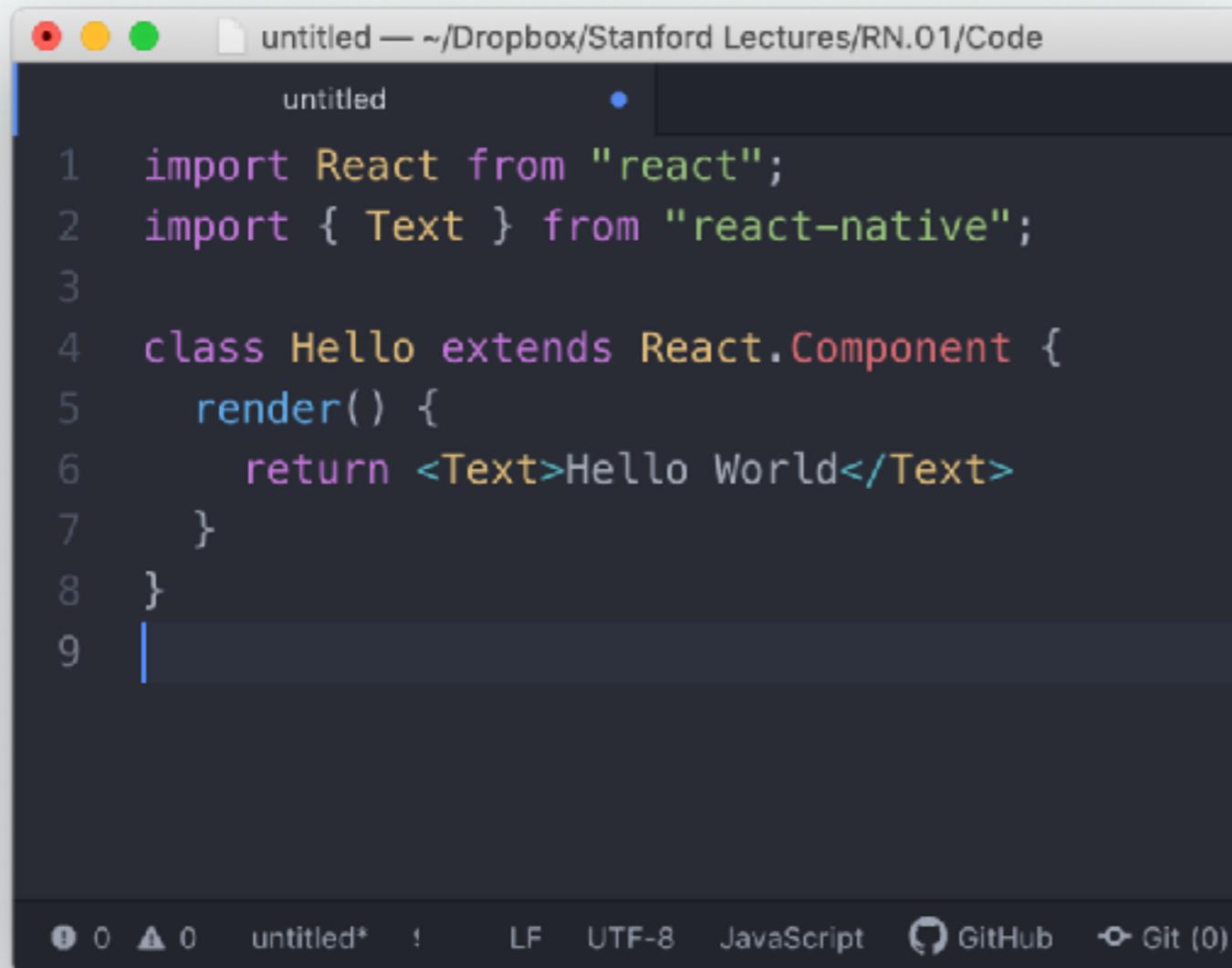
A screenshot of a code editor window titled "untitled". The code editor displays the following JavaScript code:

```
1 import React from "react";
2
3 class Hello extends React.Component {
4     render() {
5         // TODO
6     }
7 }
8
```

The code editor interface includes standard window controls (red, yellow, green) at the top left, and a status bar at the bottom with icons for file operations, a GitHub link, and a Git status showing 0 changes.

1. Import React
2. Define component

First Component



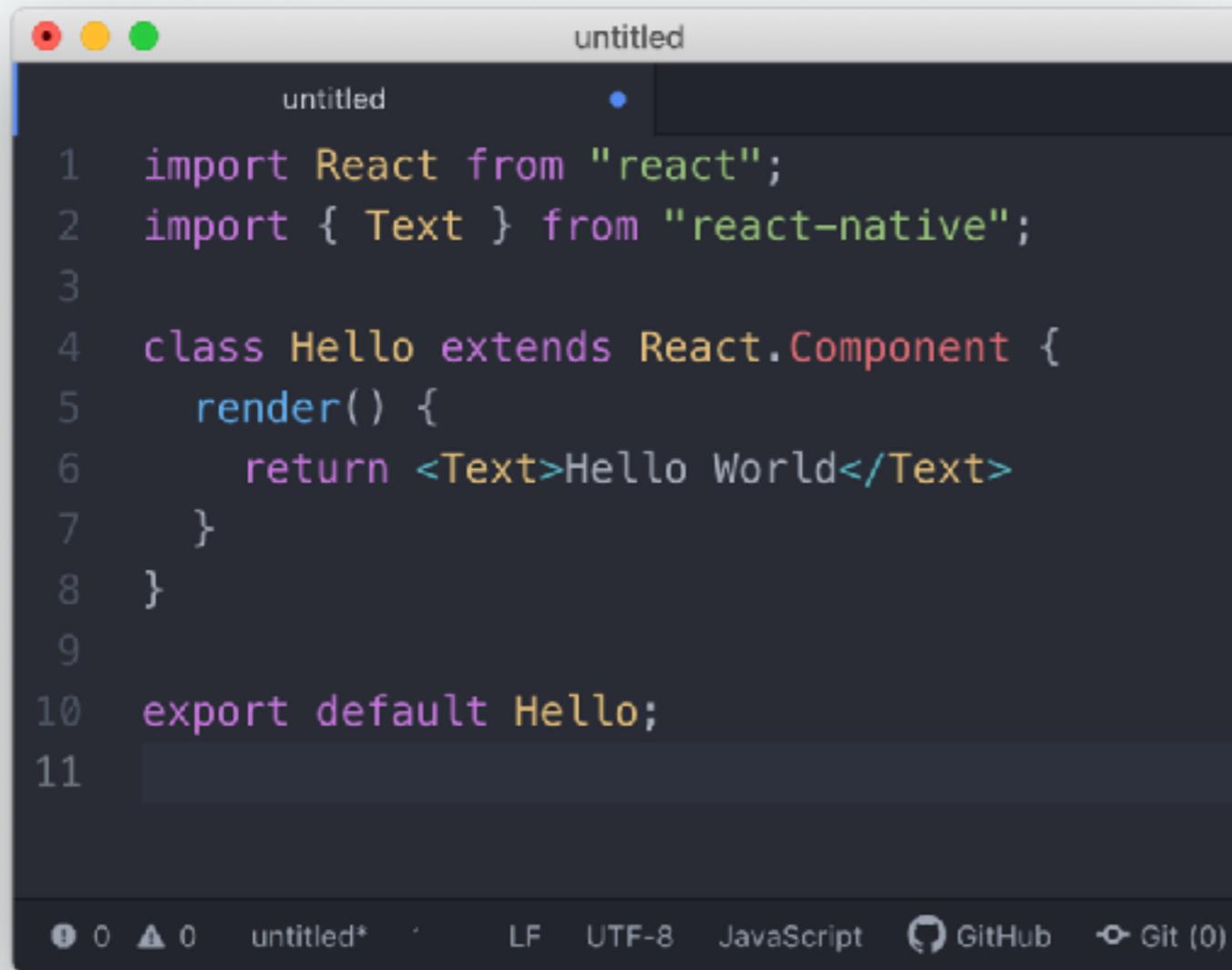
A screenshot of a terminal window titled "untitled — ~/Dropbox/Stanford Lectures/RN.01/Code". The window contains the following code:

```
1 import React from "react";
2 import { Text } from "react-native";
3
4 class Hello extends React.Component {
5   render() {
6     return <Text>Hello World</Text>
7   }
8 }
9 |
```

The terminal window also shows status icons at the bottom: 0 files, 0 errors, untitled*, LF, UTF-8, JavaScript, GitHub, and Git (0).

1. Import React
2. Define component
3. Implement render

First Component



A screenshot of a code editor window titled "untitled". The code editor displays the following JavaScript code:

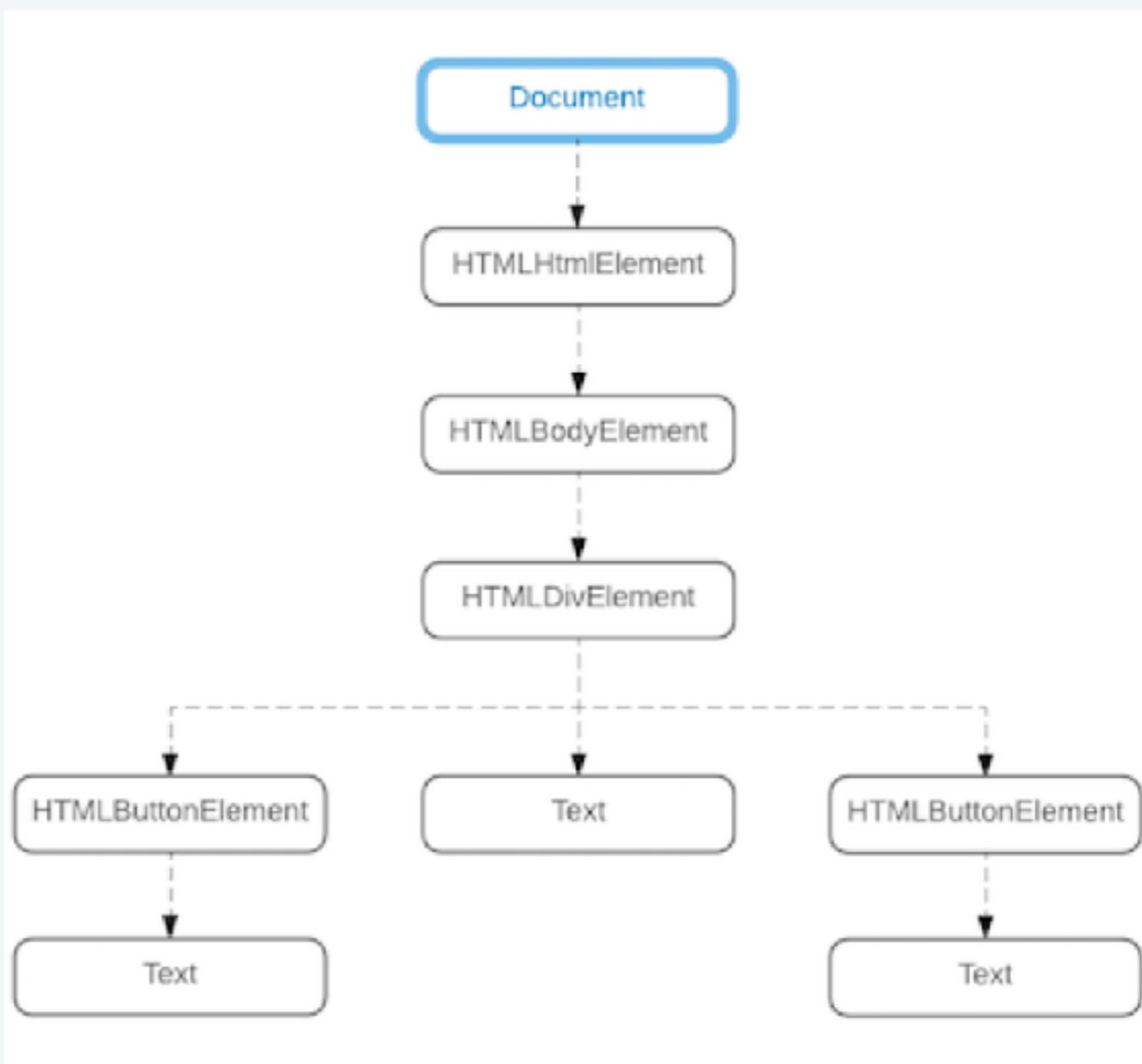
```
1 import React from "react";
2 import { Text } from "react-native";
3
4 class Hello extends React.Component {
5   render() {
6     return <Text>Hello World</Text>
7   }
8 }
9
10 export default Hello;
11
```

The status bar at the bottom shows the following information: 0 0 ▲ 0 untitled* LF UTF-8 JavaScript GitHub Git (0)

1. Import React
2. Define component
3. Implement render
4. Export component

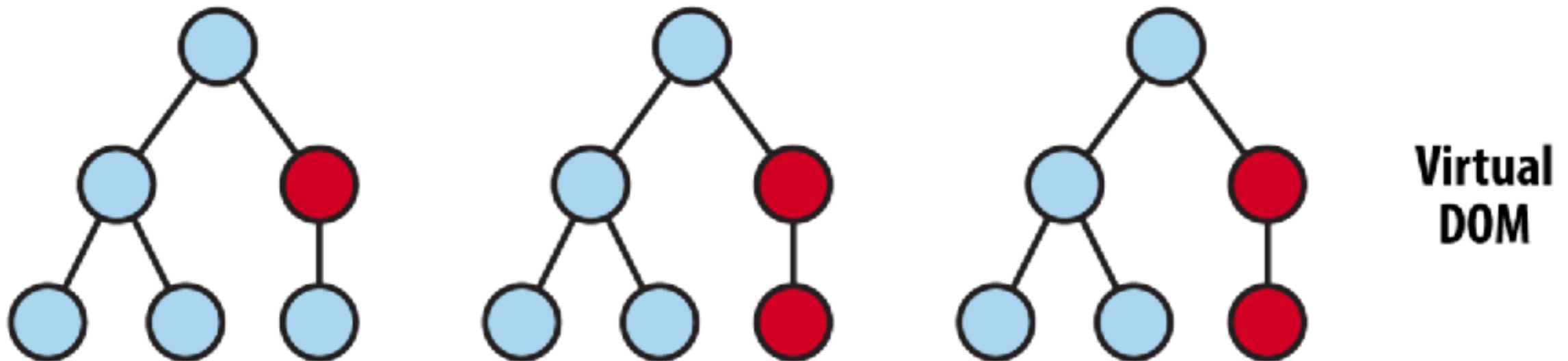
Virtual DOM (Document Object Model)

A virtual representation of a desired view state



React Native calls the render function every time a change occurs.

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start  
      <Text>Changes you make will au  
      <Text>Shake your phone to open  
    </View>  
  );  
}
```



**Virtual
DOM**

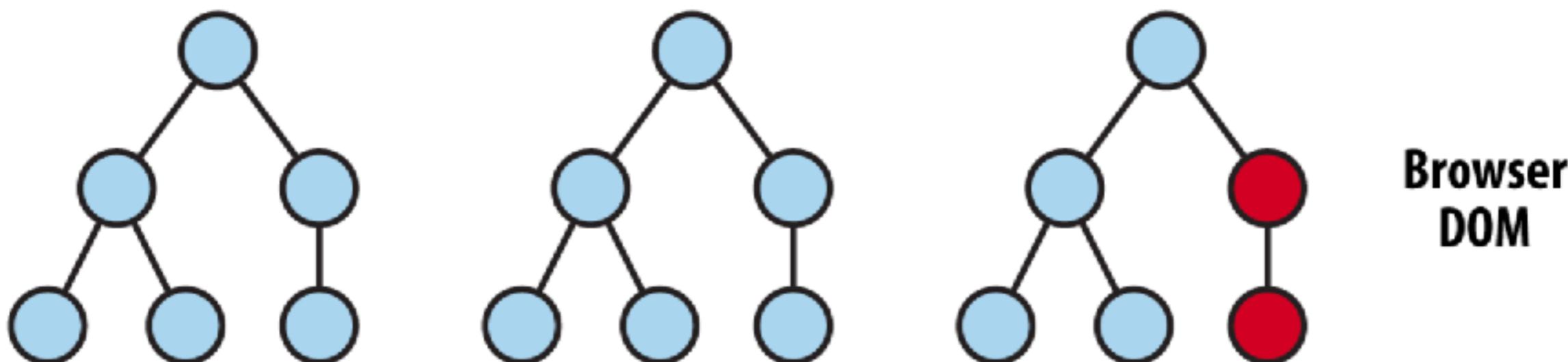
State Change



Compute Diff

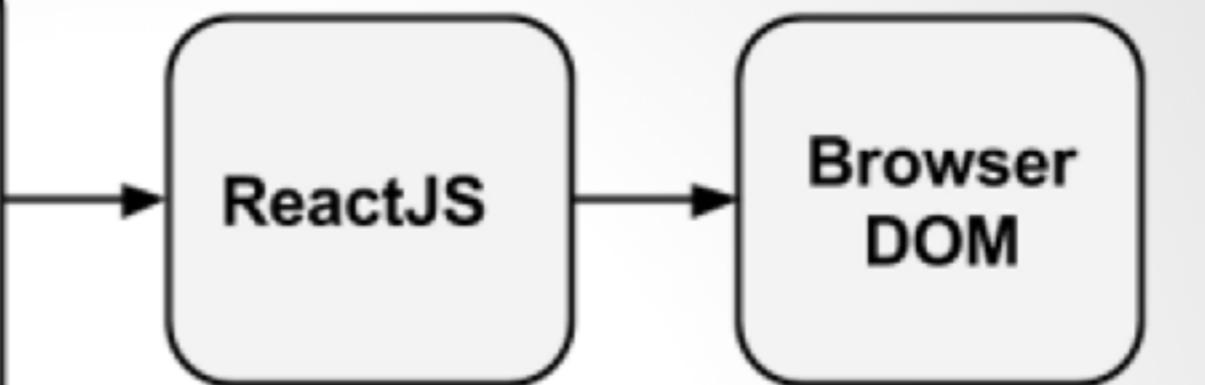


Re-render

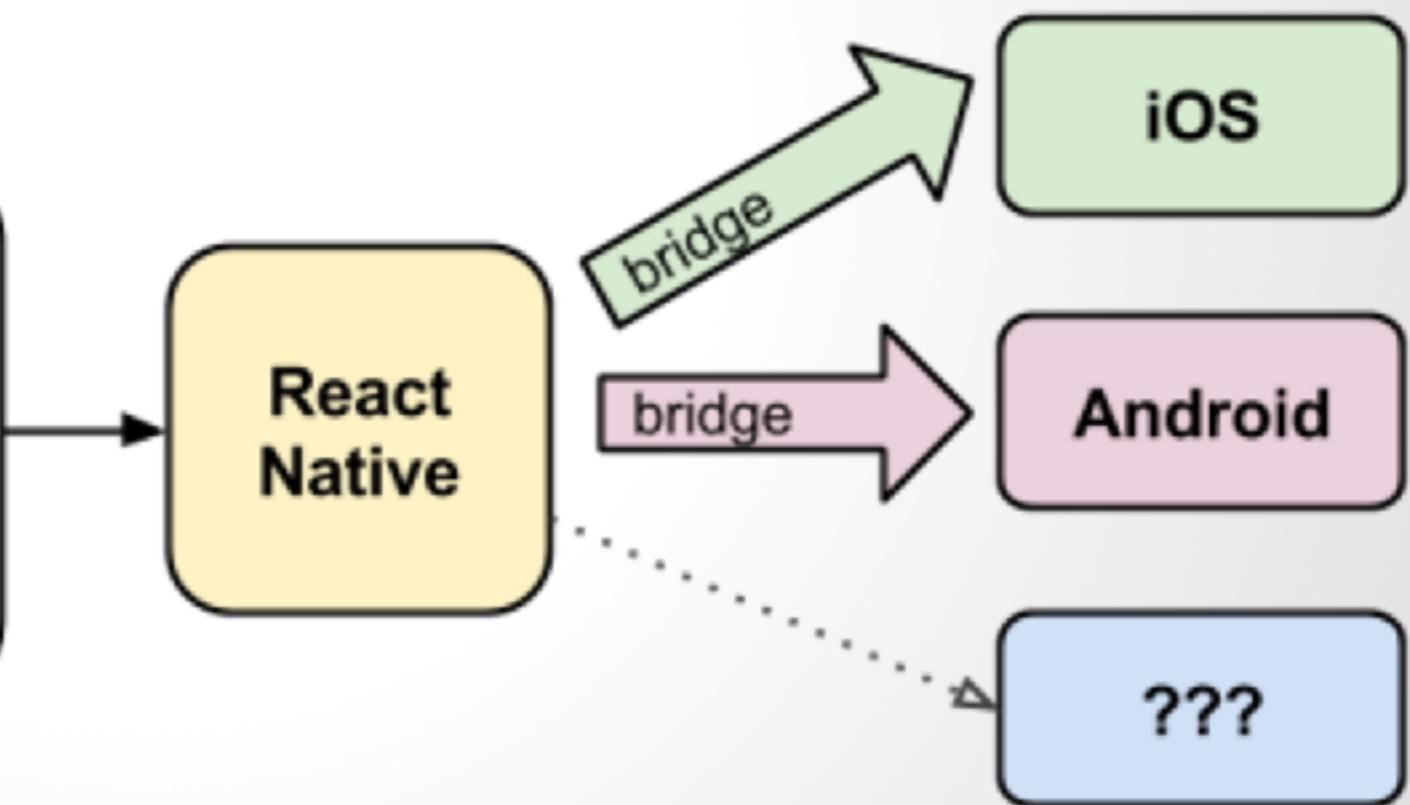


**Browser
DOM**

```
React Component
render: function() {
  return <div>Hi!</div>;
}
```



```
React Component
render: function() {
  return <View>Hi!</View>;
}
```



```
const ourNestedView = (  
  <View  
    foo='bar'>  
    <Text>42</Text>  
  </View>  
)
```



UIView



View



UILabel



TextView

Native interaction == new thread

Most of Business logic runs on JS thread

Smooth animations
UI changes

Back to project

The Props

Set by parent

Fixed true lifetime of the component

The State

When data needs to change

When **setState** is called, the App will re-render
its Component.

```
const styles = StyleSheet.create({  
  container: {  
    flex: 1,  
    backgroundColor: '#fff',  
    alignItems: 'center',  
    justifyContent: 'center',  
  },  
});
```

Basically “CSS” brought to a mobile platform

<https://github.com/vhpoet/react-native-styling-cheat-sheet>