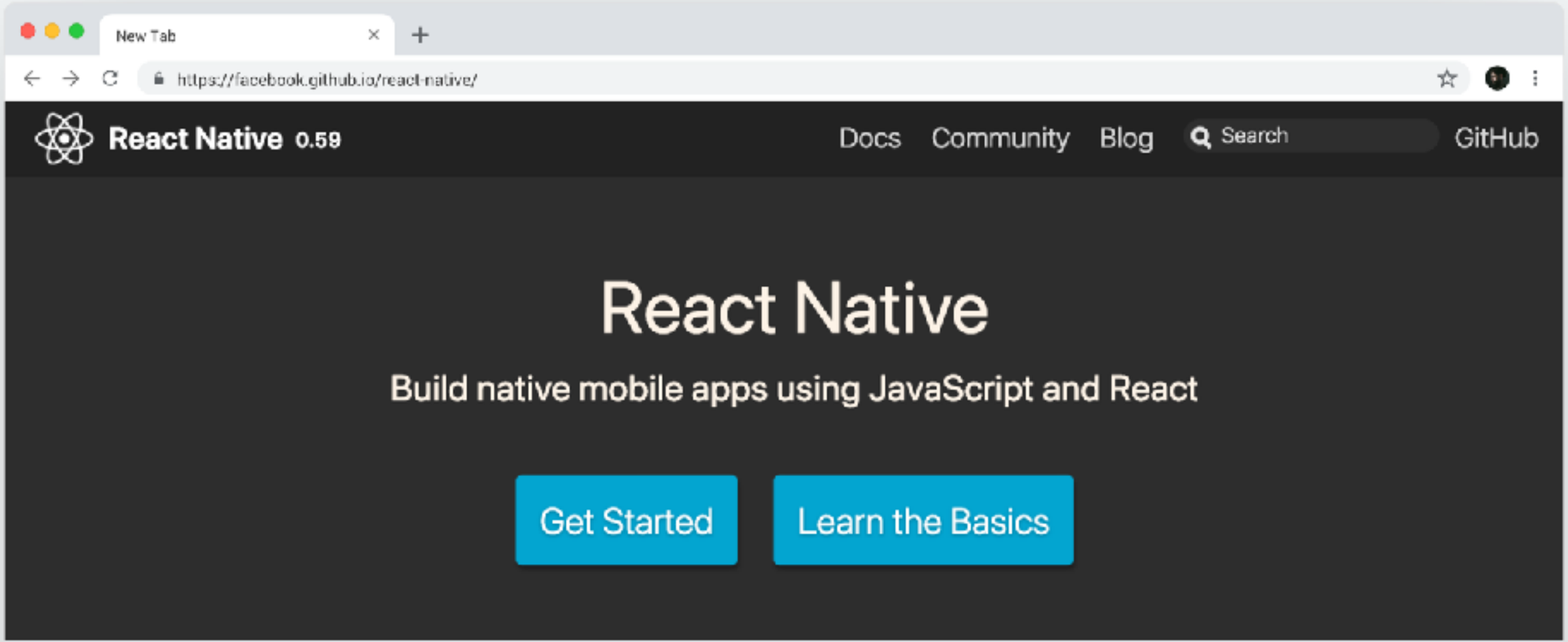
React Native 101

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:

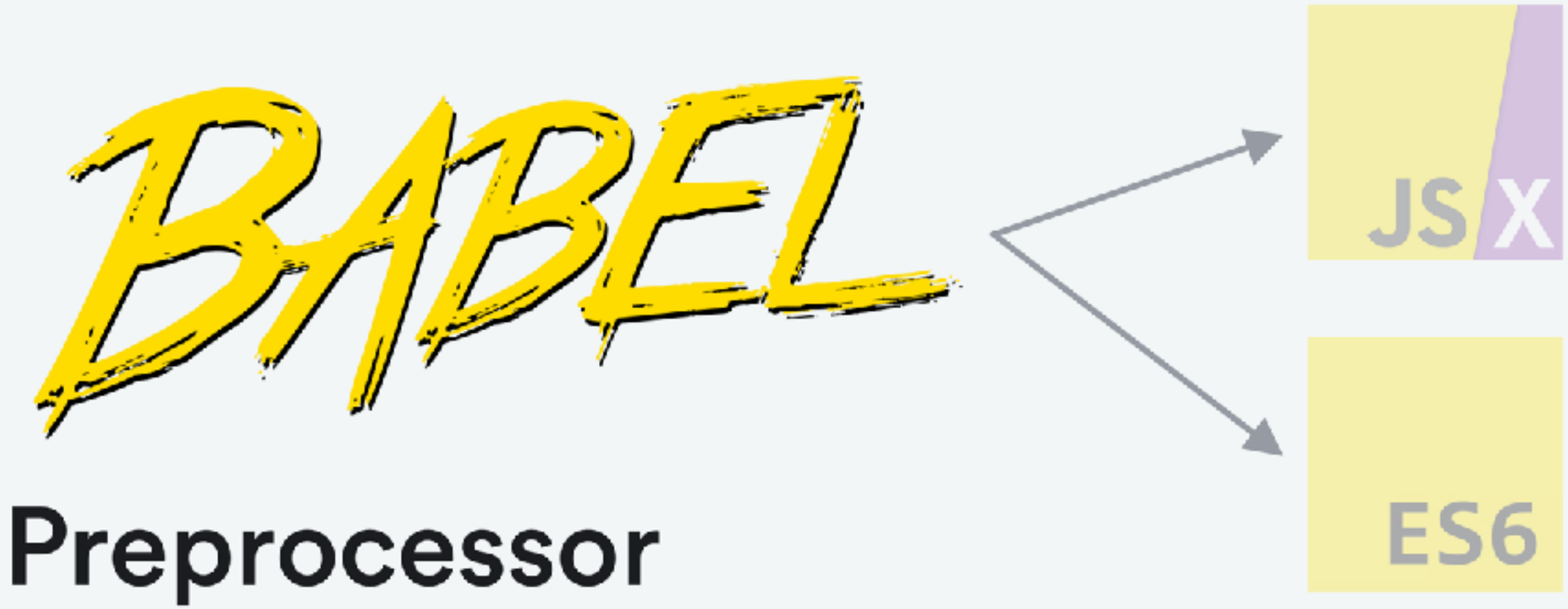
It's still improving.

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



Preprocessor

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


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


BABEL
JSX
ES6

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

```es6
const people = ['eva', 'john', 'mark']
const newPeople = [...people]
const lotsOfPeople = [...people, 'trent', 'lolita', 'james']

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

BABEL

```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))
```

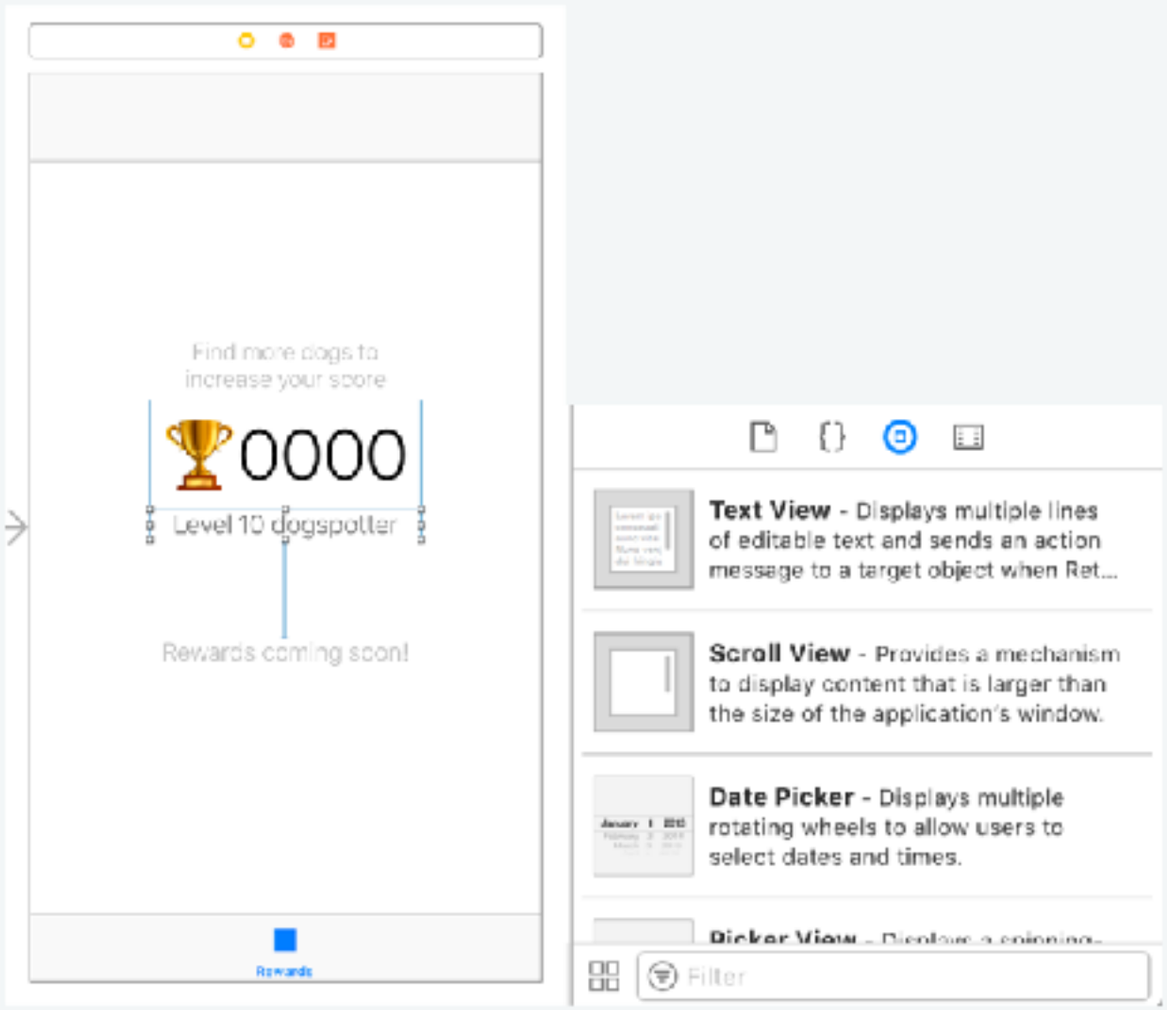
BABEL

```
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);
});
```

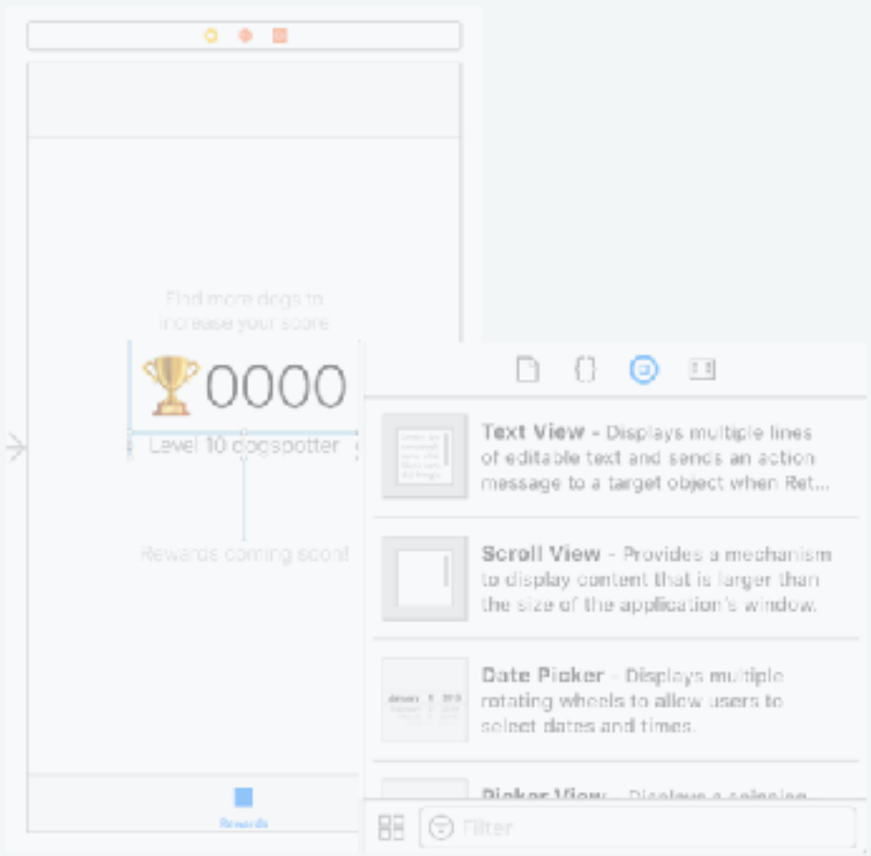
ES6 → ES7 → ES8

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

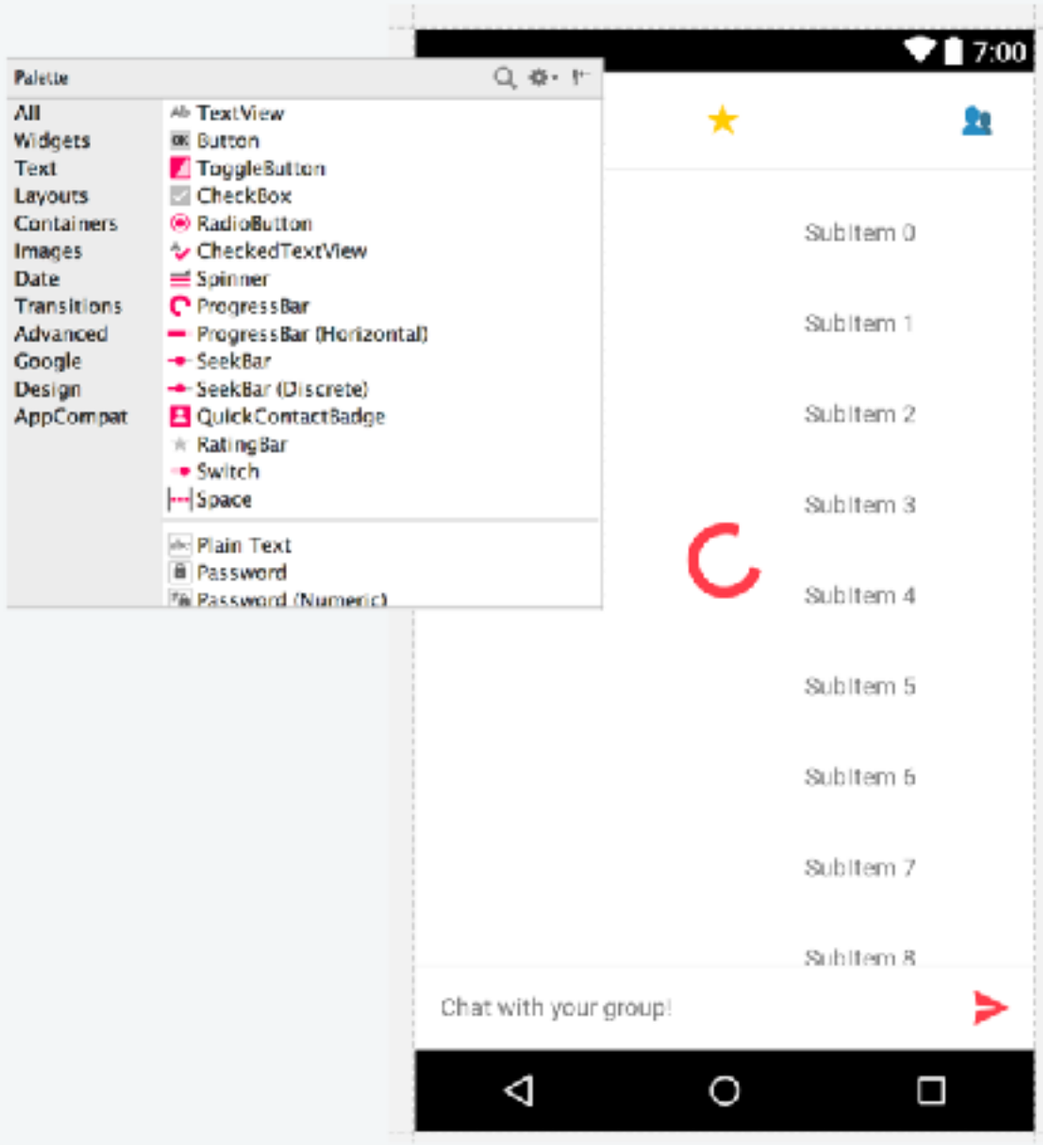

```
let textLayer = CATextLayer()
textLayer.backgroundColor = color.cgColor
textLayer.foregroundColor = UIColor.white.cgColor
textLayer.frame = frame
textLayer.alignmentMode = kCAAlignmentLeft
textLayer.isWrapped = true
let font = CTFontCreateWithName("System" as CFString, 18.0, nil)
textLayer.font =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

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



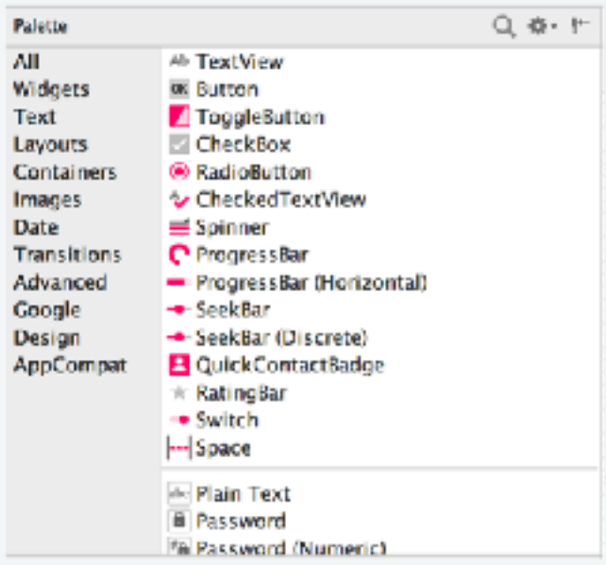

```
let textLayer = CATextLayer()
...
self.view.layer.addSublayer(textLayer)
```

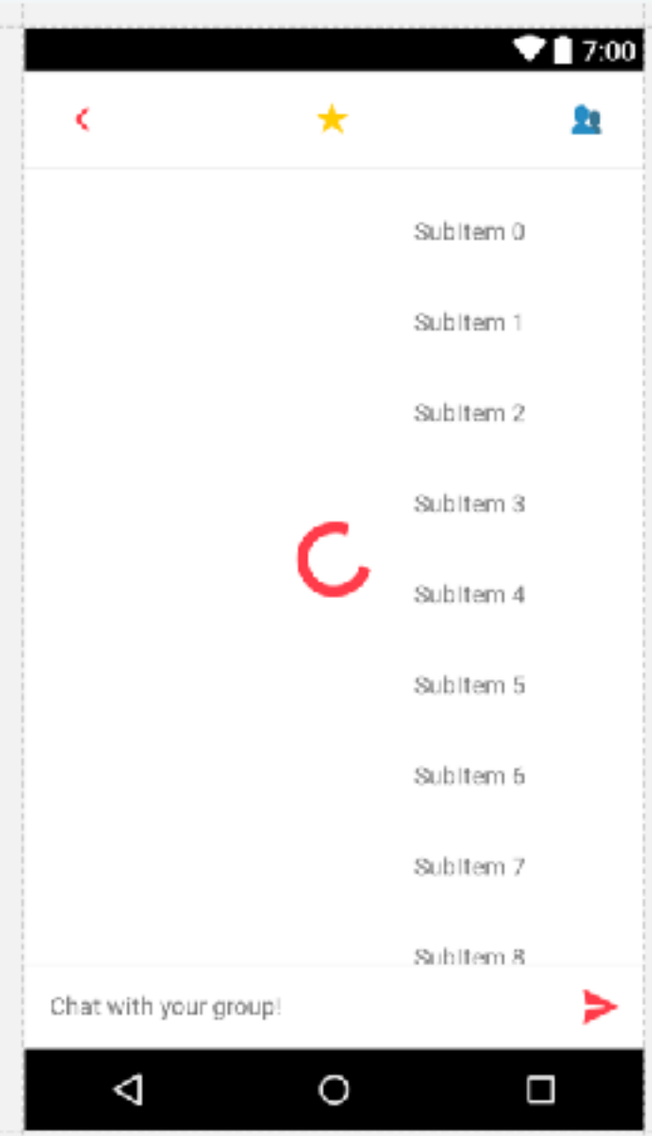


```
LinearLayout myLayout = findViewById(R.id.main);

Button myButton = new Button(this);
myButton.setLayoutParams(new
LinearLayout.LayoutParams(
LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.MATCH_PARENT));

myLayout.addView(myButton);
```

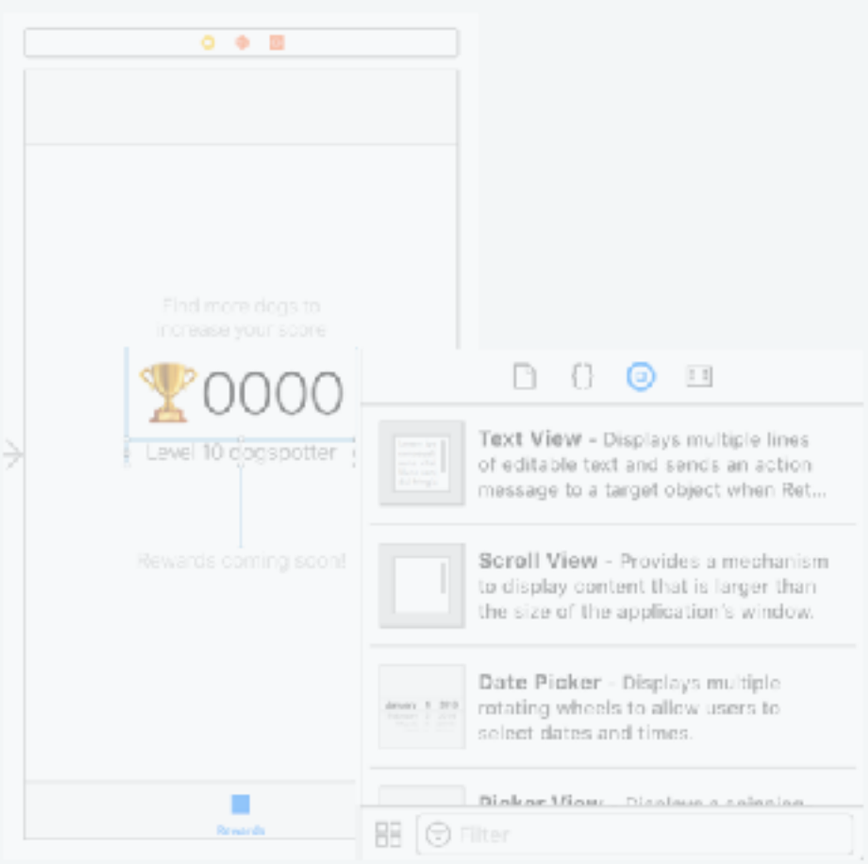


```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:gravity="center"
    android:minHeight="48dp"
    android:orientation="horizontal">

    <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)
```



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'
  )
);
```

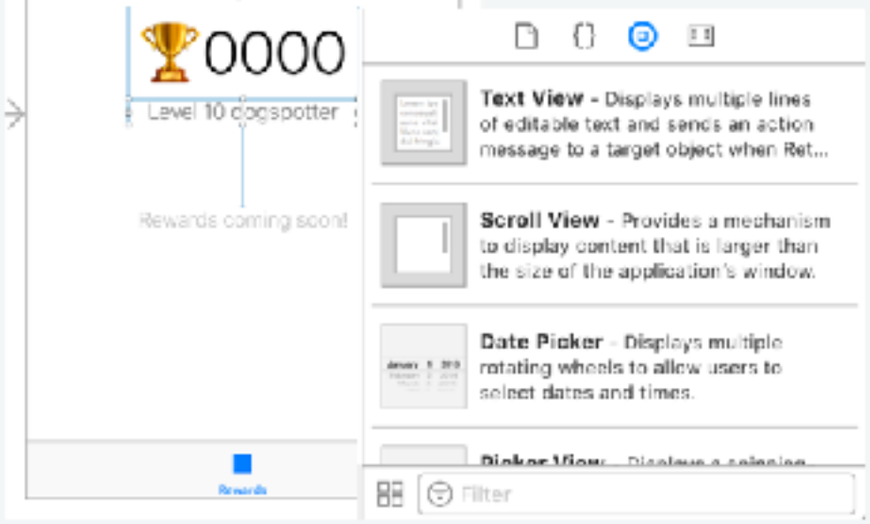
With JSX

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

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

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/edittext_bg"
    android:gravity="center_vertical"
    android:minHeight="?attr/actionBarSize"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/edittext"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_weight="1"
        android:background="@null"
        android:hint="Type a message"
        android:maxLines="5" />

    <!--<ImageButton
        <!--android:id="@+id/button_chatbox_send"
        <!--android:layout_width="wrap_content"
        <!--android:layout_height="wrap_content"
        <!--android:layout_marginRight="16dp"
        <!--android:background="?attr/selectableItemBackgroundBorderless"
        <!--android:src="@drawable/ic_send_black_24dp" />-->

</LinearLayout>
```

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



```
class Project extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.welcome}>
          Welcome to React Native!
        </Text>
        <Text style={styles.instructions}>
          To get started, edit index.ios.js
        </Text>
        <Text style={styles.instructions}>
          Press Cmd+R to reload,{'\n'}
```

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

Setup

You need JavaScript in your terminal

Setup

You need a dependency manager

or

Setup

or

Install React Native + Deps

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

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

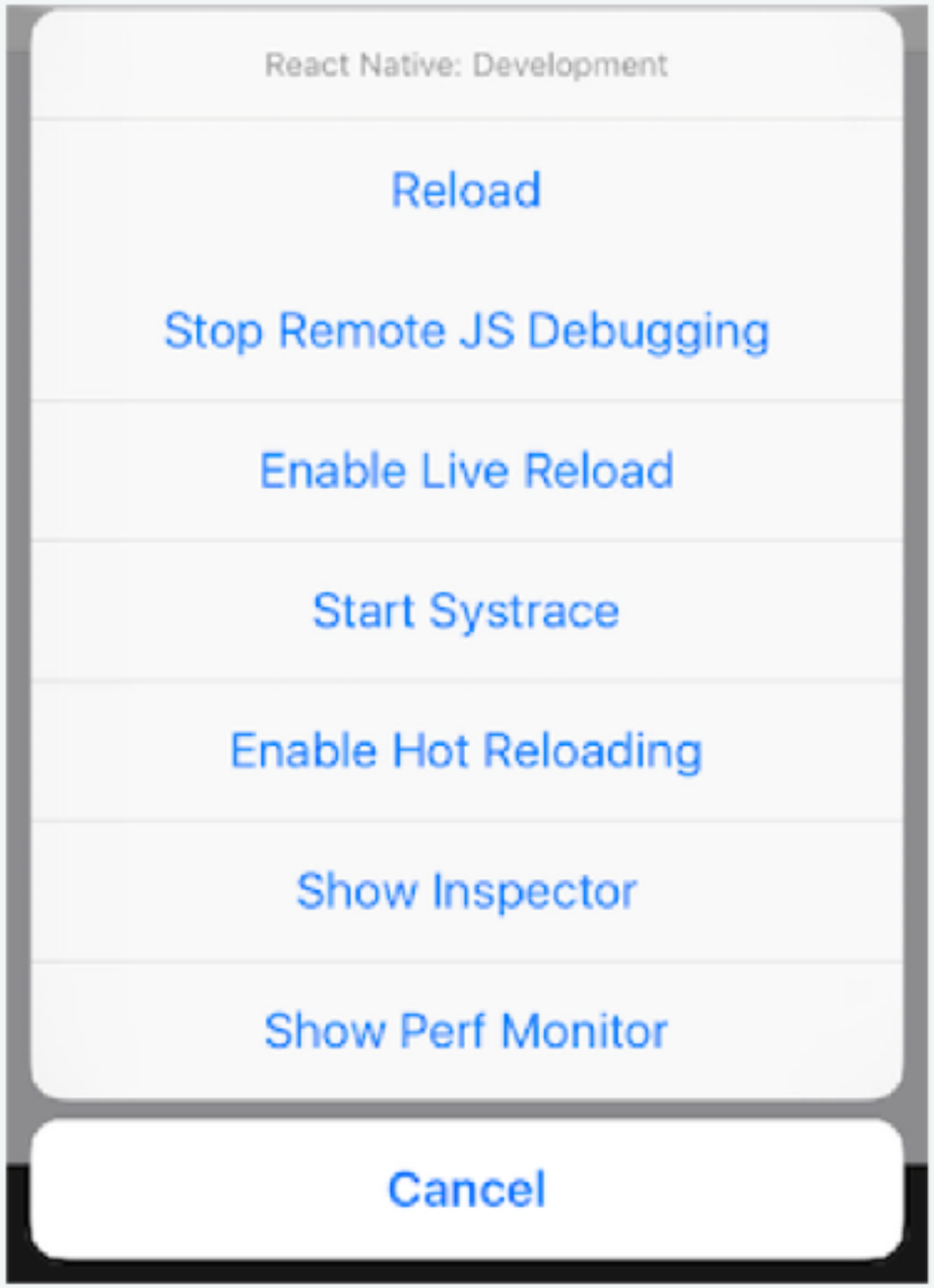
https://facebook.github.io/react-native/docs/getting-started

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

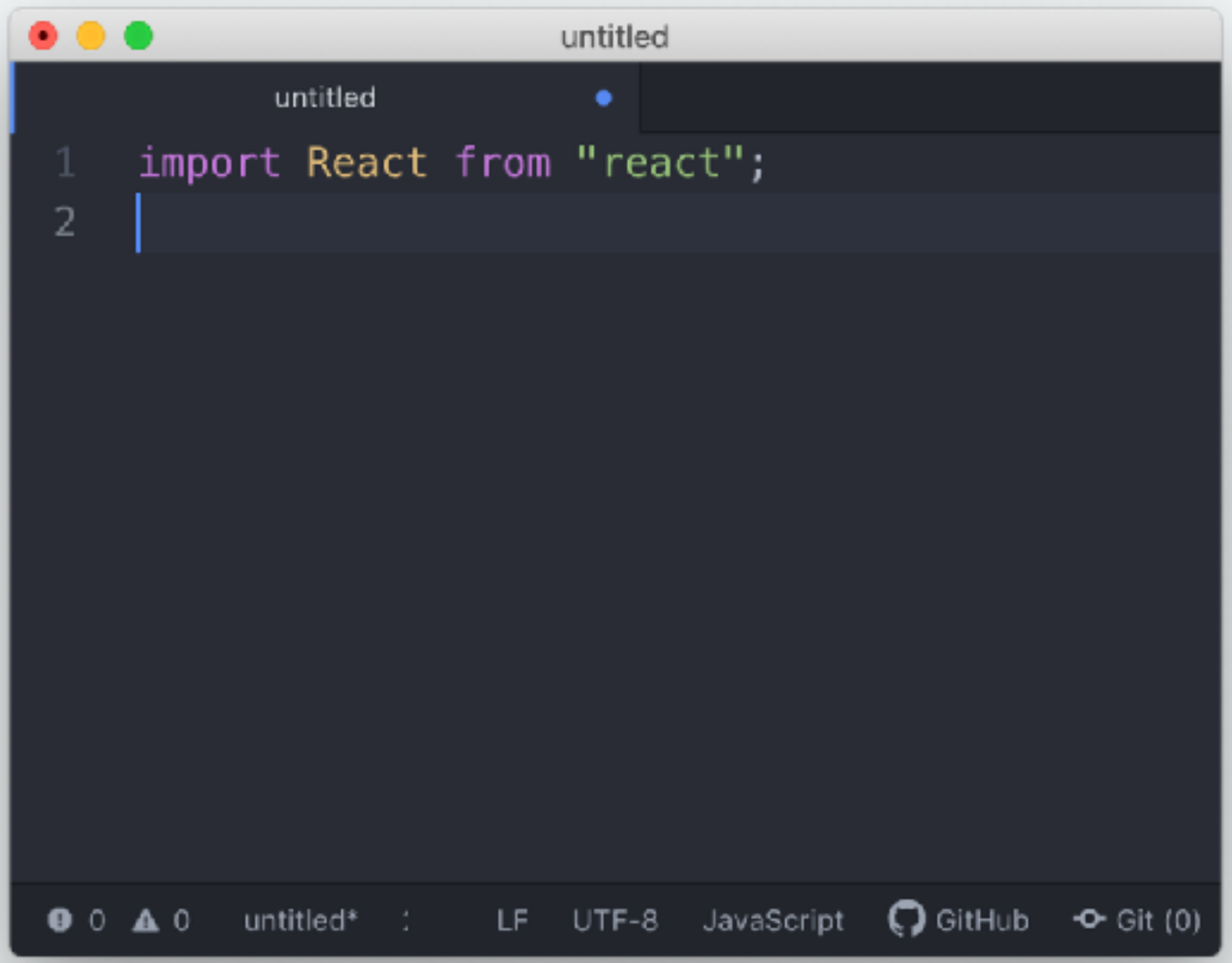
`console.error()`

`console.warn()`

`IS_TESTING`

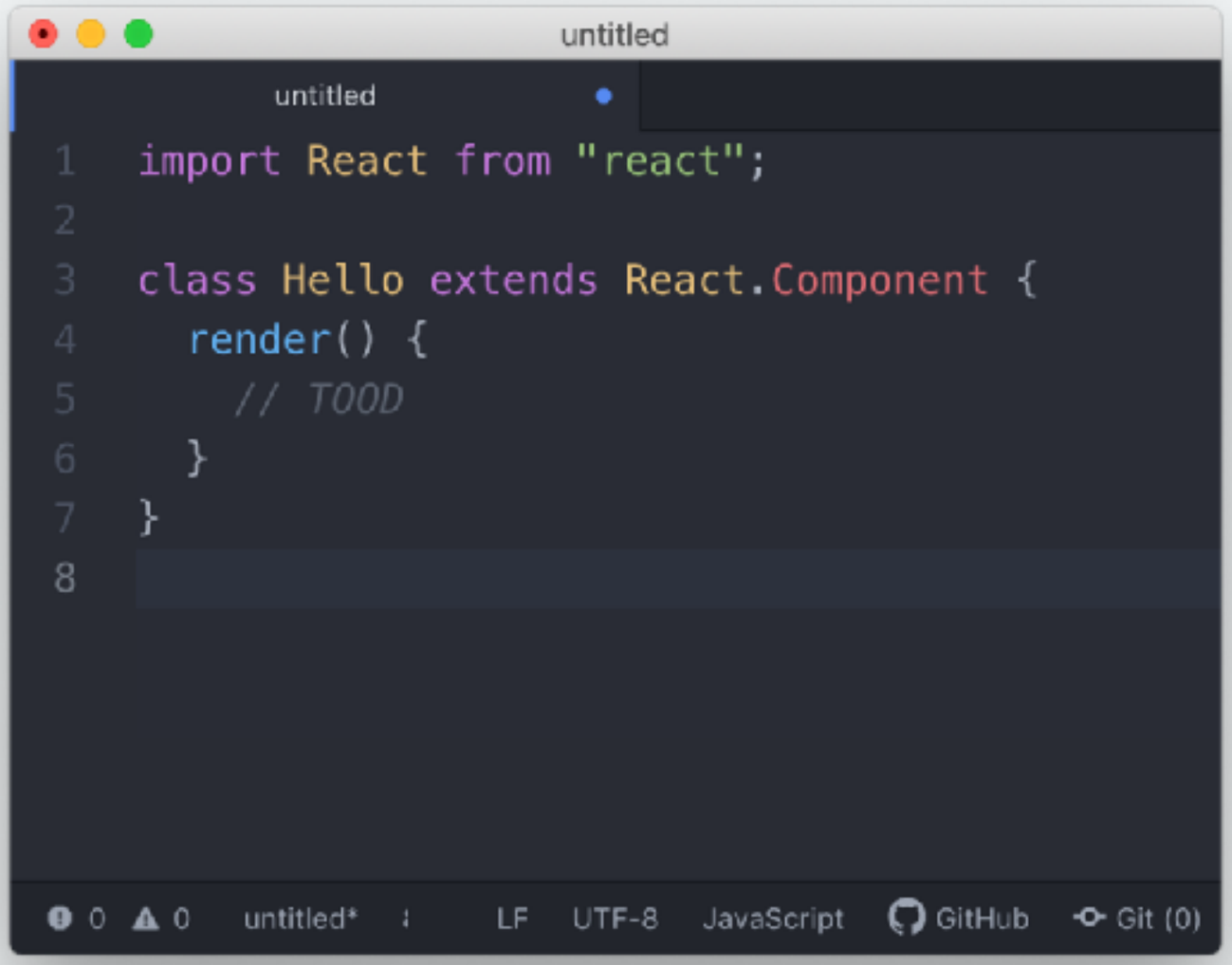
RedBoxes and **YellowBoxes** are automatically disabled in **release** (production) builds.

First Component



1. Import React

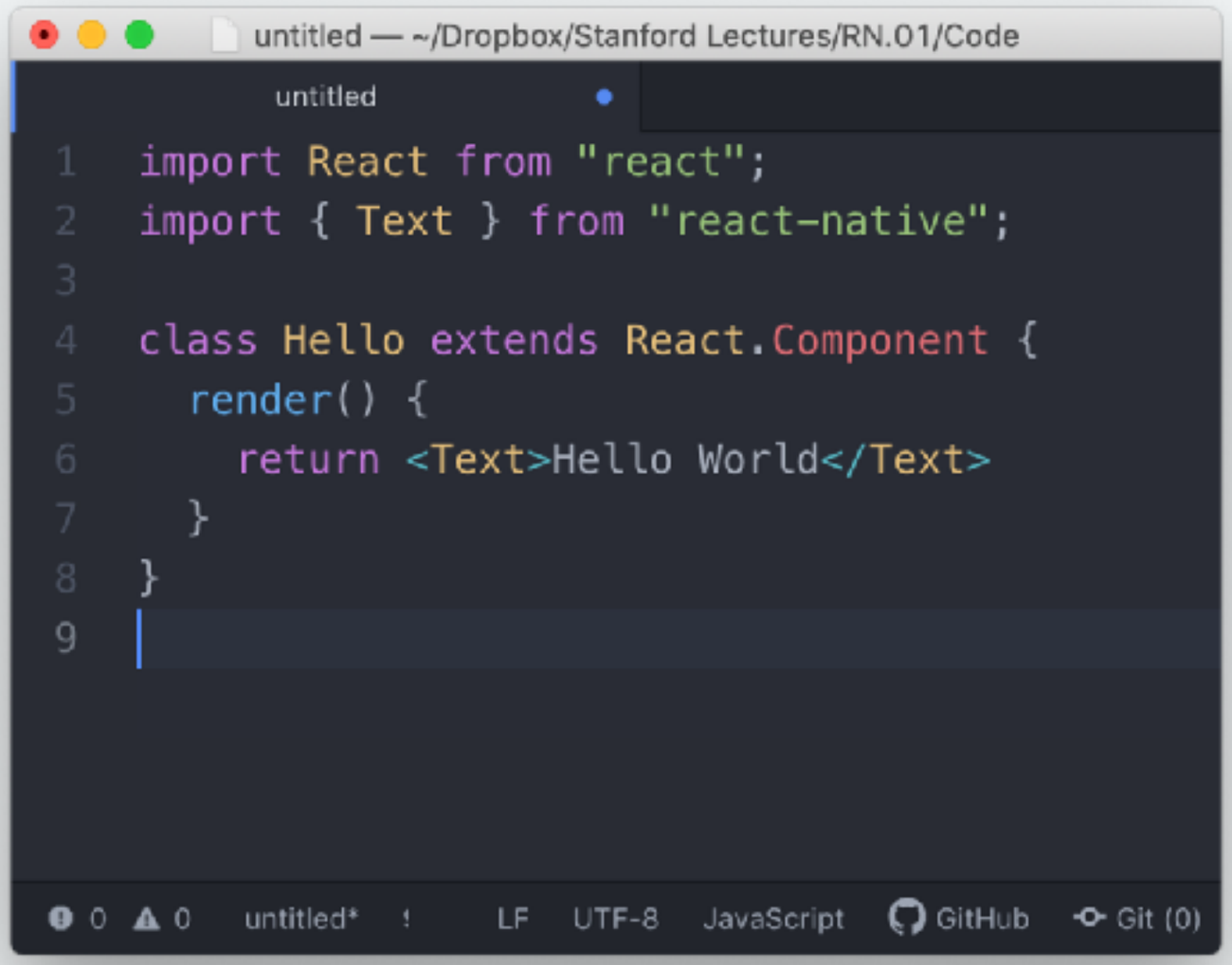
First Component



1. Import React

2. Define component

First Component

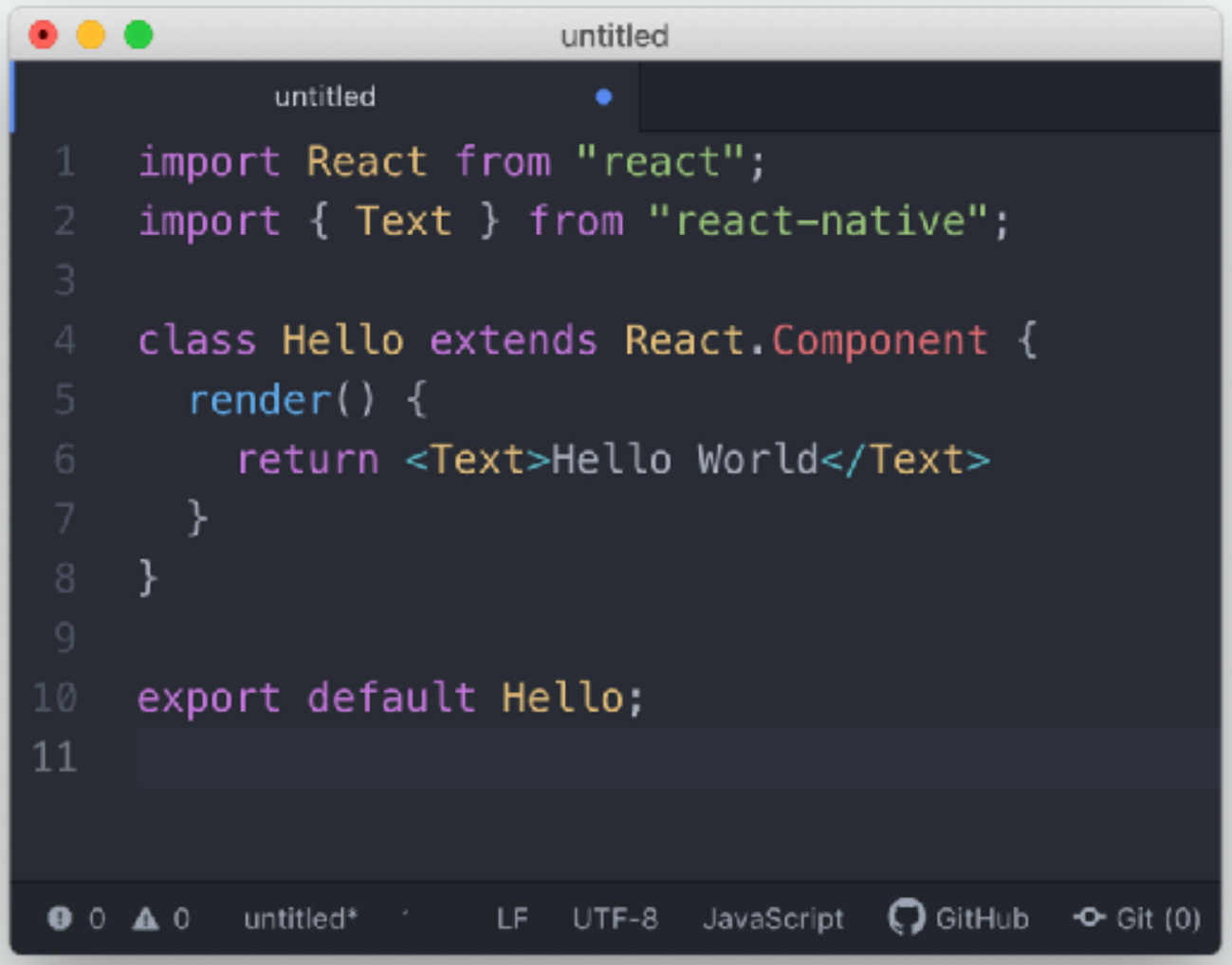


1. Import React

2. Define component

3. Implement render

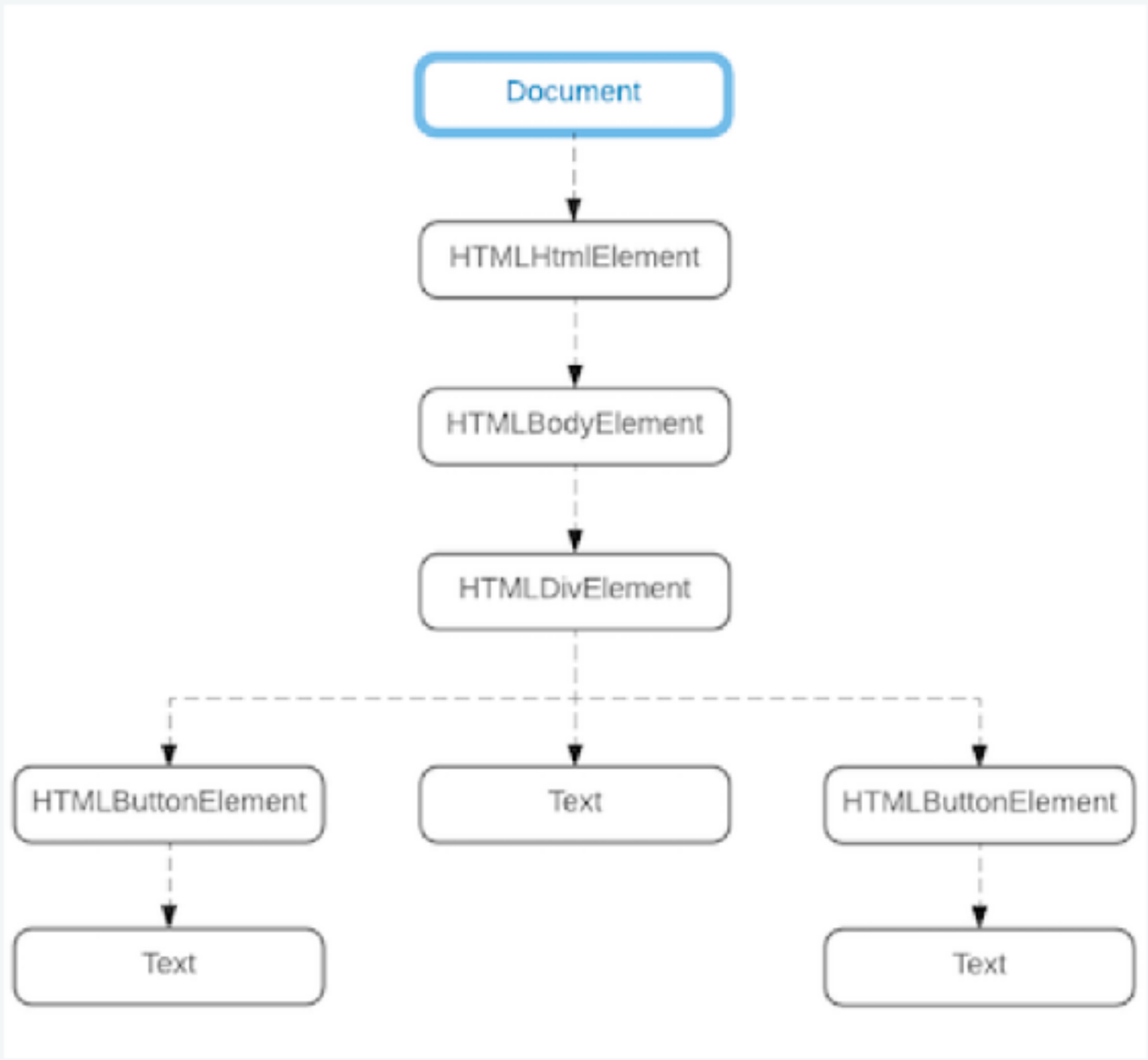
First Component



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

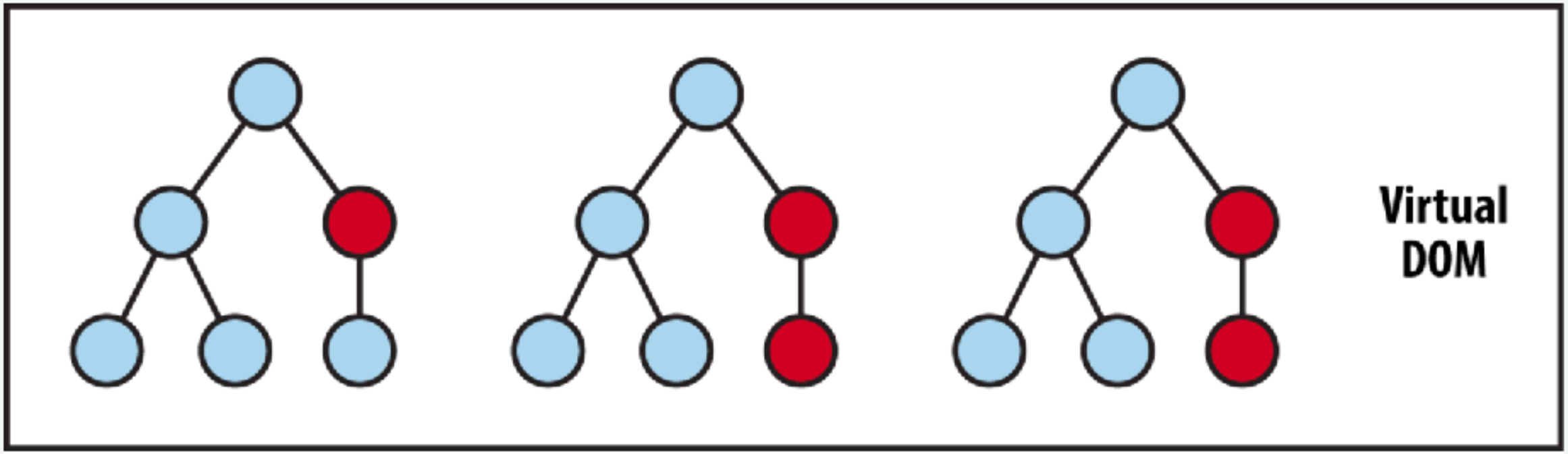
Virtual DOM (Document Object Model)

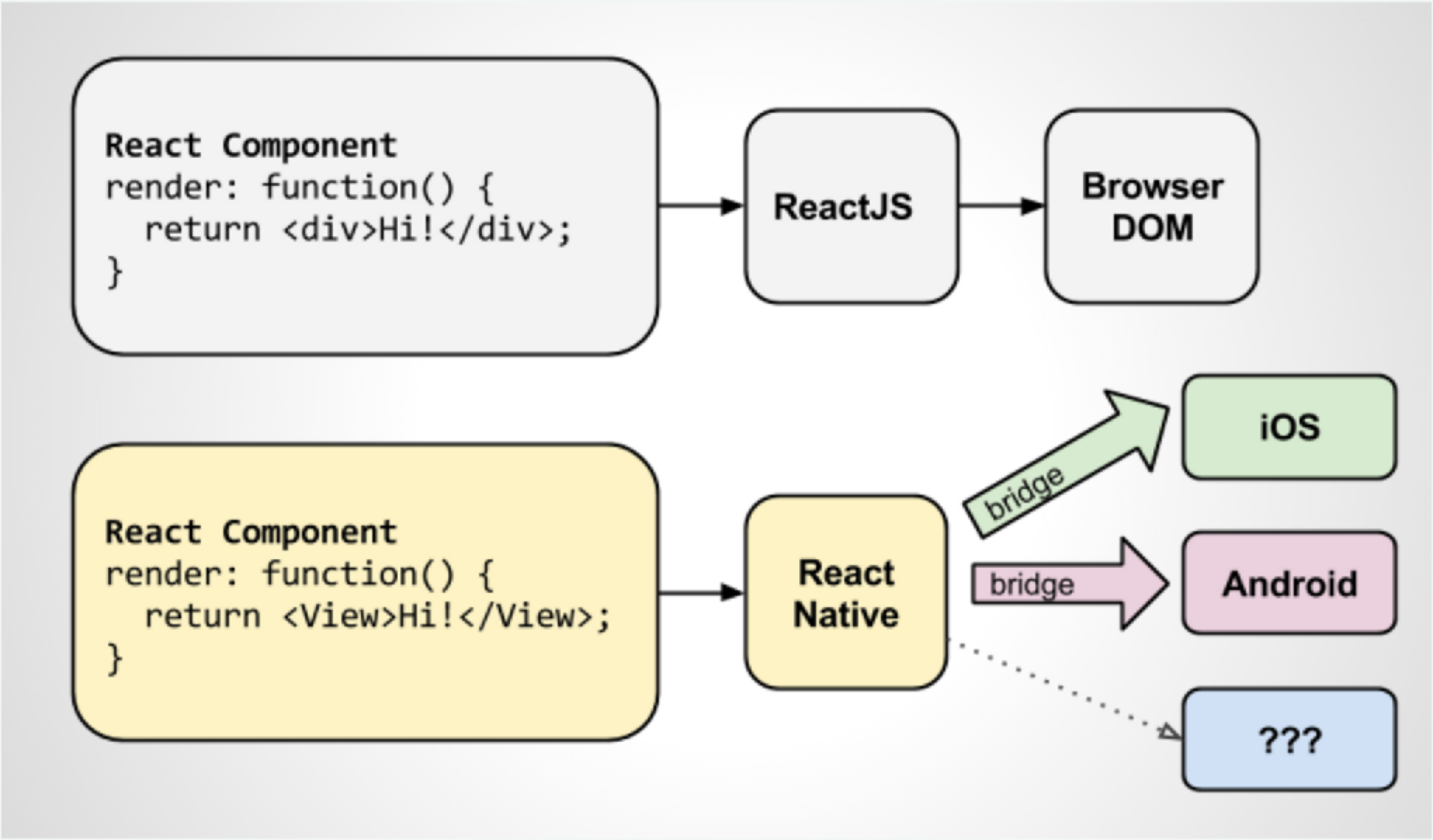
A virtual representation of a desired view state



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 oper
      </View>
    );
}
```


Virtual
DOM
State Change
Compute Diff
Re-render
Browser
DOM


React Component
render: function() {
return Hi! ;
}
ReactJS
Browser DOM
React Component
render: function() {
return <View>Hi!</View>;
}
React Native
bridge
iOS
bridge
Android
???

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

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 tru 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