

CS47: Cross-Platform Mobile Development

Lecture 1B: Introduction to Javascript (ES6)

<https://cs-47.stanford.edu>



cs47-fall19.slack.com

James Landay
Abdallah AbuHashem
Tiffany Manuel
Cisco Vlahakis
Vy Mai

Fall 2019

CS 47 ▾

● Abdallah AbuHashem

Jump to...

Threads

Channels

class

general

random

🔒 staff

+ Add a channel

Direct Messages

♥ Slackbot

● Abdallah AbuHashem (y...

👤 Alyssa Romanos

👤 Avni Kakkar

👤 Cisco Vlahakis

👤 Cynthia Jia

👤 David

👤 Gabby Delos Reyes

👤 Gabby Delos Reyes, Tiffa...

👤 Shashank Rammoorthy

👤 Vy

#general

☆ | 👤 16 | 📄 0 | Company-wide announcements and ...

🔍 Search

@ ☆ ⋮

#general

@Vy created this channel on September 12th. This is the very beginning of the #general channel. Purpose: This channel is for workspace-wide communication and announcements. All members are in this channel. (edit)

+ Add an app 👤 Add people to this channel

Thursday, September 12th

Vy 6:23 PM

joined #general along with 13 others.

Tuesday, September 24th

Lucy Zhu 2:12 PM

Does anyone know where are we supposed to submit assignment 1?

👤 1 reply 1 day ago




Kevin Lin 8:00 PM

joined #general along with Olayinka A.

📎 Message #general @ 😊

Join our
Slack
<https://tinyurl.com/cs47slack2019>

Cross-Platform Mobile Development

Overview	
Schedule	 17
Readings	

Overview

This course teaches the fundamentals of cross-platform mobile application development with a focus on the React Native framework (RN). The goal is to help students develop best practices in creating apps for both iOS and Android by using Javascript and existing web + mobile development paradigms. Students will explore the unique aspects that made RN a primary tool for mobile development within Facebook, Instagram, Walmart, Tesla, and UberEats.

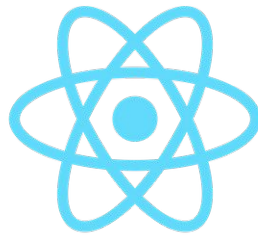
COURSE LOGISTICS

Date/Time	T/Th 10:30PM - 12:00PM
Enrollment	Please apply here and show up to the first class to enroll in the class.
Location	Wallenberg 124 (160-124)
Units	2 Pass/Fail
Instructors	Abdallah Abuhashem (aabuhash@stanford.edu) Tiffany Manuel (manuel14@stanford.edu) Vy Mai (vmmai2@stanford.edu) Cisco Vlahakis (vlahakis@stanford.edu)
Faculty Sponsor	James Landay (landay@stanford.edu)
Staff email	reactnative@cs.stanford.edu
Office hours	TBD
Prerequisites	CS 106A/B
Explore courses	CS47

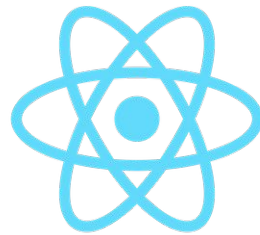
<https://cs47.stanford.edu>

Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

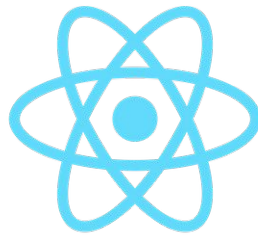


Why do you want to take this class?



Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

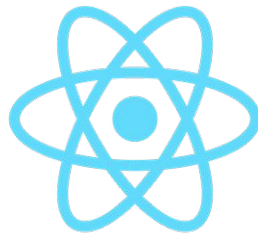


Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

Overview for today

- Introduce JavaScript Basics
- Introduce Babel
- Introduce JSX
- Create an application from scratch
 - Breakdown React Native project files

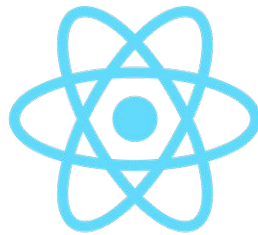


Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

Overview for today

- Introduce JavaScript Basics
- Introduce Babel
- Introduce JSX
- Create an application from scratch
 - Breakdown React Native project files



<https://es6console.com/>

JS: Variables

In ES6, variables:

- Don't have types
 - But must be declared before using them
-
- Global variables

```
var x = 1;
```

```
if (x === 1) {  
  var x = 2;  
  console.log(x);  
  // expected output: 2  
}
```

```
console.log(x);  
// expected output: 2
```

The logo for ES6 (ECMAScript 2015) is displayed on a yellow square background. The text "ES6" is in a bold, dark grey, sans-serif font.

JS: Variables

In ES6, variables:

- Don't have types
 - But must be declared before using them
-
- Local variables

```
let x = 1;
```

```
if (x === 1) {  
  let x = 2;  
  console.log(x);  
  // expected output: 2  
}
```

```
console.log(x);  
// expected output: 1
```

A yellow square containing the text "ES6" in a bold, dark grey, sans-serif font.

JS: Variables

In ES6, variables:

- Don't have types
- But must be declared before using them

- Const variables

```
const x = 1;
```

```
if (x === 1) {  
  x = 2;  
  //Error  
  console.log(x);  
  // expected output: 2  
}
```

```
console.log(x);  
// expected output: 1
```

A yellow square containing the text "ES6" in a bold, black, sans-serif font.

JS: Variables

	Global Variables	Local Variables	Constant Variables
Use	The scope here is the function in which it's declared	The scope here is the block in which it is declared	The scope here is the block, but it cannot be changed in value
Syntax	<code>var x = 10;</code>	<code>let x = 10;</code>	<code>const x = 10;</code>

A yellow square containing the text "ES6" in a bold, black, sans-serif font.

ES6

JS: If statements



ES6

JS: If statements

JS needs special care with equality.

If in doubt use `===` and `!==` (or use them always)

- Example

```
if (a > 0) {  
    return "positive";  
} else if (a === 0) {  
    return "It's a zero";  
} else {  
    return "NOT positive";  
}
```

The logo for ES6 (ECMAScript 2015) is displayed on a yellow square background. The text "ES6" is in a bold, black, sans-serif font.

JS: Loops

You have many options for loops in JS

- For loops

```
for (var i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}
```

- For of loops

```
for (var element of arr) {  
  console.log(arr);  
}
```

A yellow square containing the text "ES6" in a bold, dark grey sans-serif font.

ES6

JS: Loops

You have many options for loops in JS

- For each loops

```
arr.forEach(function(element) {  
  console.log(element);  
});
```

- While loops

```
While (true) {  
  console.log("You can't stop me");  
}
```

A yellow square containing the text "ES6" in a bold, dark grey sans-serif font.

ES6

JS: Functions

- Functions in JS are declared in the following way

```
function addition(a, b = 10) {  
  return a + b;  
}
```

A yellow square containing the text "ES6" in a bold, dark grey sans-serif font.

ES6

JS: Functions

- Functions in JS are declared in the following way

```
function addition(a, b = 10) {  
  return a + b;  
}
```

- Another way is as follows

```
var addition = (a, b = 10) => {  
  return a + b;  
}
```

A yellow square containing the text "ES6" in a bold, black, sans-serif font.

JS: Functions

- Functions in JS can turn passed in arguments to arrays

```
function addition(a, ...b) {  
  return a + b.length;  
}  
console.log(addition(2,1,7,5));  
// 2 + 3 = 5
```

- On the opposite side, we can do

```
function addition(a, b, c) {  
  return a + b + c;  
}  
console.log(addition(...[1,2,3]));  
// 1 + 2 + 3 = 6
```

The logo for ES6 (ECMAScript 2015) is displayed on a yellow square background. It consists of the letters "ES" in a bold, sans-serif font, followed by a large, stylized number "6".

ES6

JS: Objects

- Objects are similar to dictionaries and/or maps in other languages

```
let obj = {  
  name: 'John',  
  age: 17,  
};  
console.log(obj.name + ' ' + obj['age']);
```

- Other ways of representing properties

```
let obj = {  
  ['full' + 'name']: 'John Doe',  
  //same as age: age  
  age,  
  lorem() {  
    return "ipsum";  
  },  
};
```

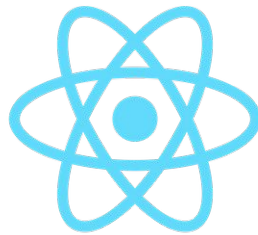
The logo for ES6 (ECMAScript 2015) is displayed on a yellow square background. The text "ES6" is in a bold, dark grey, sans-serif font.

Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

Overview for today

- Introduce JavaScript Basics ✓
- Introduce JSX
- Create an application from scratch
 - Breakdown React Native project files

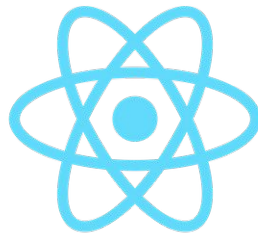


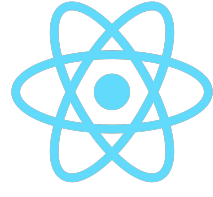
Last lecture

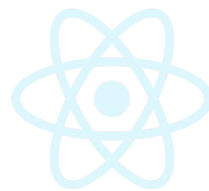
- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

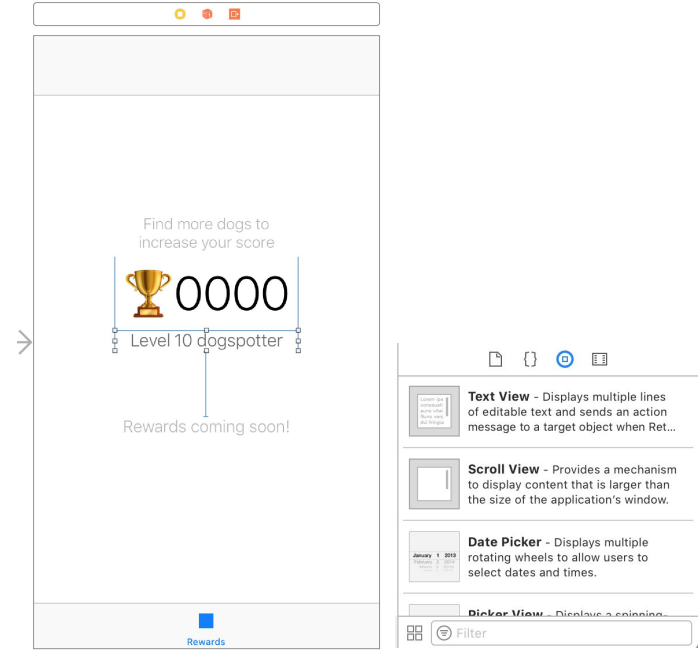
Overview for today

- Introduce JavaScript Basics ✓
- Introduce JSX
- Create an application from scratch
 - Breakdown React Native project files



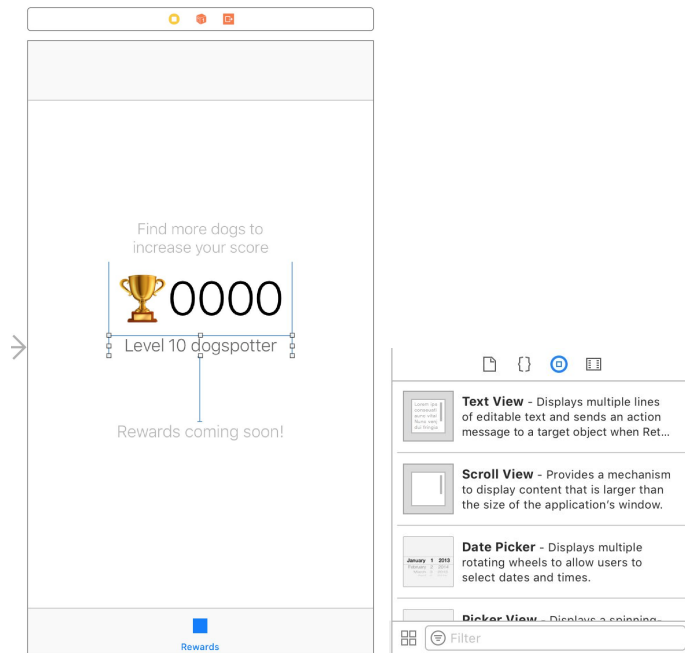








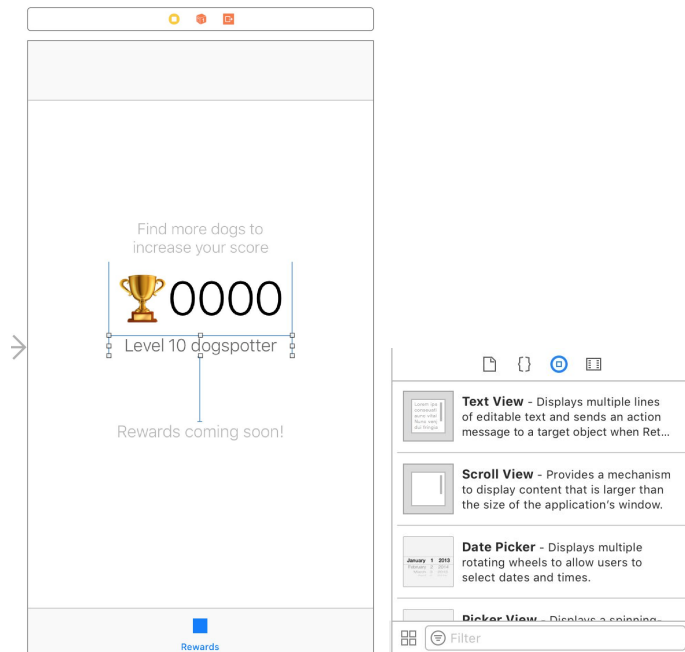
Unfortunately you will have to deal
with autolayout constraints

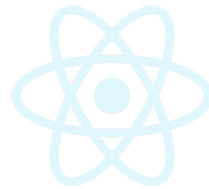




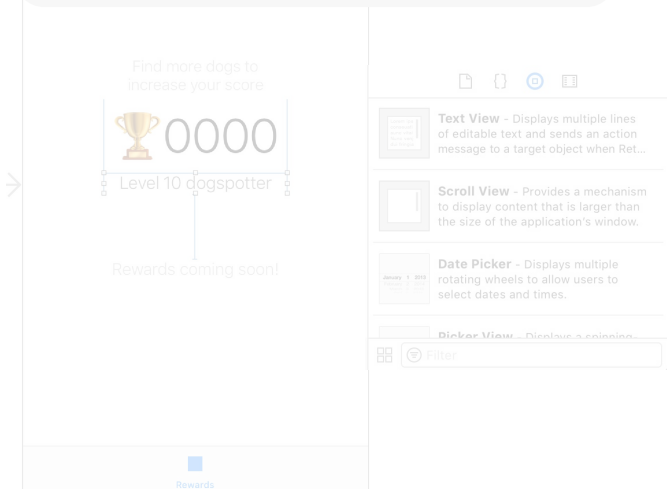
```
let textLayer = CATextLayer()
textLayer.backgroundColor = color.cgColor
textLayer.foregroundColor = UIColor.white.cgColor
textLayer.frame = frame
textLayer.alignmentMode = kCAAlignmentLeft
textLayer.isWrapped = true
let font = CTFontCreateWithName("System" as CFString, 18.0, nil)
textLayer.font =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



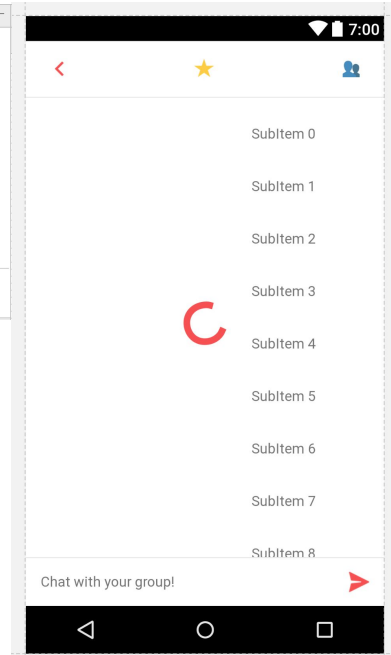
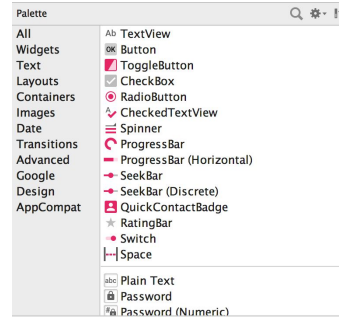


```
LinearLayout myLayout = findViewById(R.id.main);
```

```
Button myButton = new Button(this);
```

```
myButton.setLayoutParams(new LinearLayout.LayoutParams(  
    LinearLayout.LayoutParams.MATCH_PARENT,  
    LinearLayout.LayoutParams.MATCH_PARENT));
```

```
myLayout.addView(myButton);
```



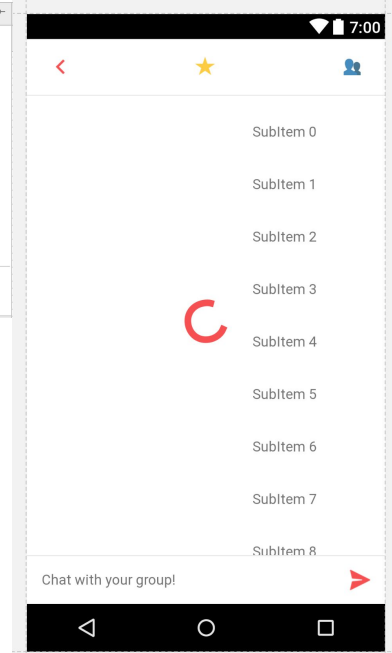
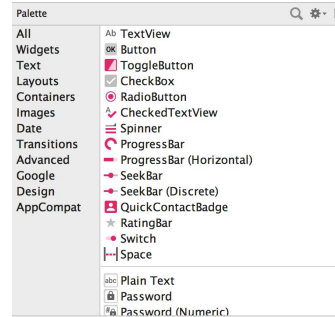


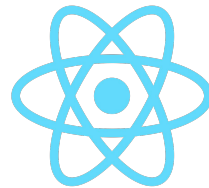
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:gravity="center"
    android:minHeight="48dp"
    android:orientation="horizontal">

    <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)
```

Find more dogs to
increase your score



Rewards coming soon!

Rewards



Text View - Displays multiple lines of editable text and sends an action message to a target object when Ret...

Scroll View - Provides a mechanism to display content that is larger than the size of the application's window.

Date Picker - Displays multiple rotating wheels to allow users to select dates and times.

DatePickerView - Displays a date picker.



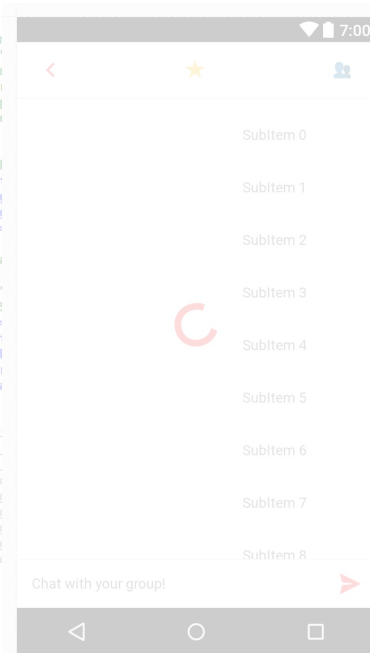
Filter

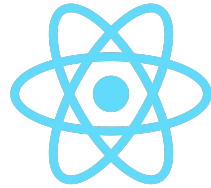
```
<LinearLayout  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:background="@android:color/white"  
  android:gravity="center"  
  android:minHeight="48dp"  
  android:orientation="horizontal"
```

```
<EditText  
  android:id="@+id/chat_input"  
  android:layout_width="match_parent"  
  android:layout_height="40dp"  
  android:layout_weight="1"  
  android:background="@android:color/white"  
  android:ems="10"  
  android:enabled="true"  
  android:hint="Chat with your group!"  
  android:inputType="textMultiLine"  
  android:maxLines="4"  
  android:paddingEnd="16dp"  
  android:paddingLeft="16dp"  
  android:paddingRight="16dp"  
  android:paddingStart="16dp"  
  android:textAppearance="@style/TextAppearance.AppCompat.EditText"
```

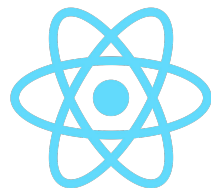
```
<!-- ImageButton -->  
<!-- android:id="@+id/send" -->  
<!-- android:layout_width="wrap_content" -->  
<!-- android:layout_height="wrap_content" -->  
<!-- android:background="@android:color/white" -->  
<!-- android:padding="16dp" -->  
<!-- android:paddingLeft="16dp" -->  
<!-- android:paddingRight="16dp" -->  
<!-- android:paddingStart="16dp" -->  
<!-- android:paddingEnd="16dp" -->  
<!-- android:src="@android:drawable/ic_dialog_send" -->
```

```
</LinearLayout>
```



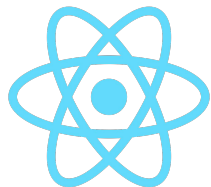


How do we build layouts in React Native?



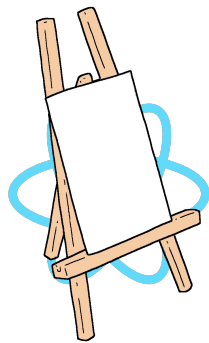
How do we build layouts in React Native?

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



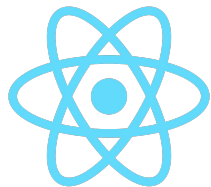
Originally, the process of creating a View using React looked like this:

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



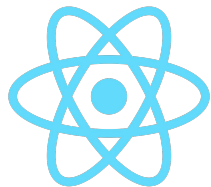
Originally, the process of creating a View using React looked like this:

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



Originally, the process of nesting Text within a View using React looked like this:

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



Originally, the process of nesting Text within a View using React looked like this:

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



JSX

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>  
)
```



JSX

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

No JSX

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

JSX is a shortcut for using the `React.createElement()` API

JSX Benefits

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

- UI has a clear hierarchical structure. What you see in code mirrors what you will get.

JSX Benefits

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

- UI has a clear hierarchical structure. What you see in code mirrors what you will get.



JSX Benefits

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

- UI has a clear hierarchical structure. What you see in code mirrors what you will get.
- This makes it easier for designers to contribute to code.

JSX Benefits

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

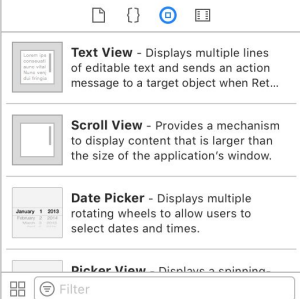
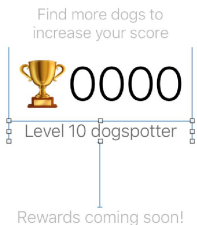
- UI has a clear hierarchical structure. What you see in code mirrors what you will get.
- This makes it easier for designers to contribute to code.
- You get the accessibility of templates AND the power of JS.



```
let textLayer = CATextLayer()
```

• • •

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

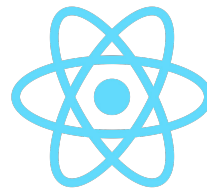
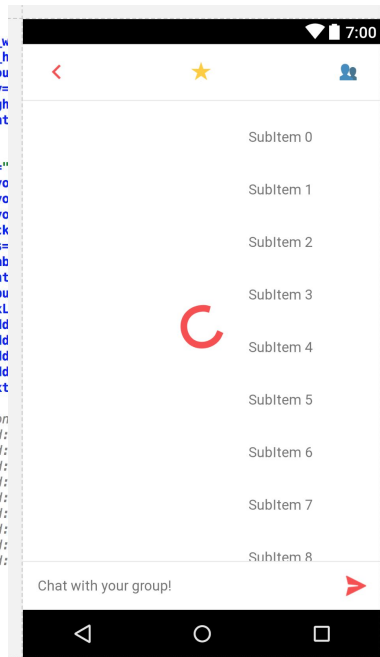


```
<LinearLayout
    android:layout_w
    android:layout_h
    android:backgrou
    android:gravity=
    android:minHeigh
    android:orientat
```

```
<EditText  
    android:id="@+id/  
    android:layout_  
    android:layout_  
    android:backgro  
    android:ems=10  
    android:enablen  
    android:hint="E  
    android:inputTy  
    android:maxLen  
    android:paddin  
    android:paddin  
    android:paddin  
    android:paddin  
    android:text=""
```

[illegible]

</LinearLayout>



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

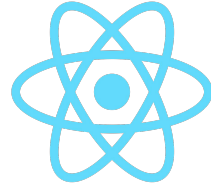


Image — deco

Display an image, either from a remote URI or bundled as a static asset.

CORE UI REACT-NATIVE



Map View — deco

Display a native map with optional annotations and overlays.

CORE UI REACT-NATIVE

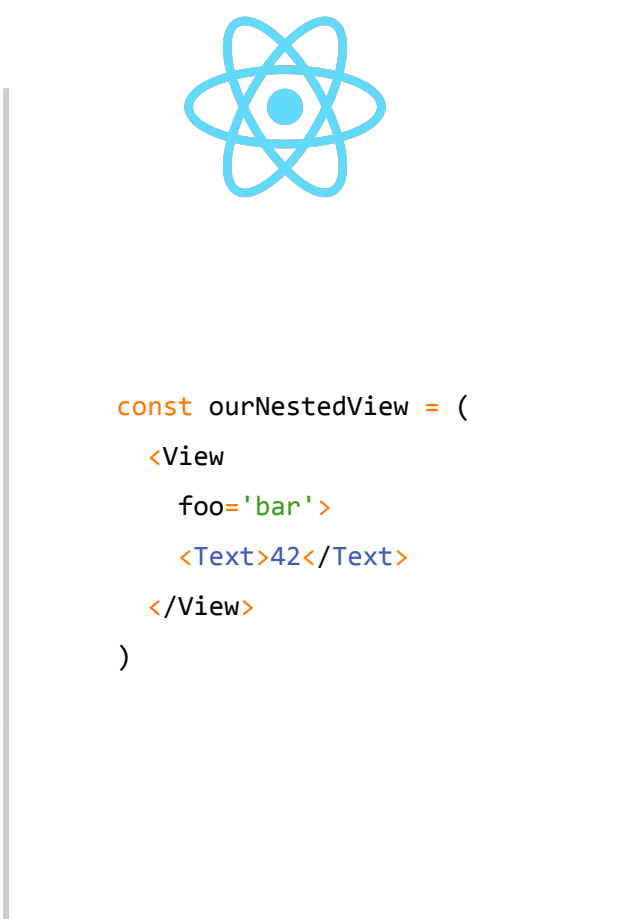
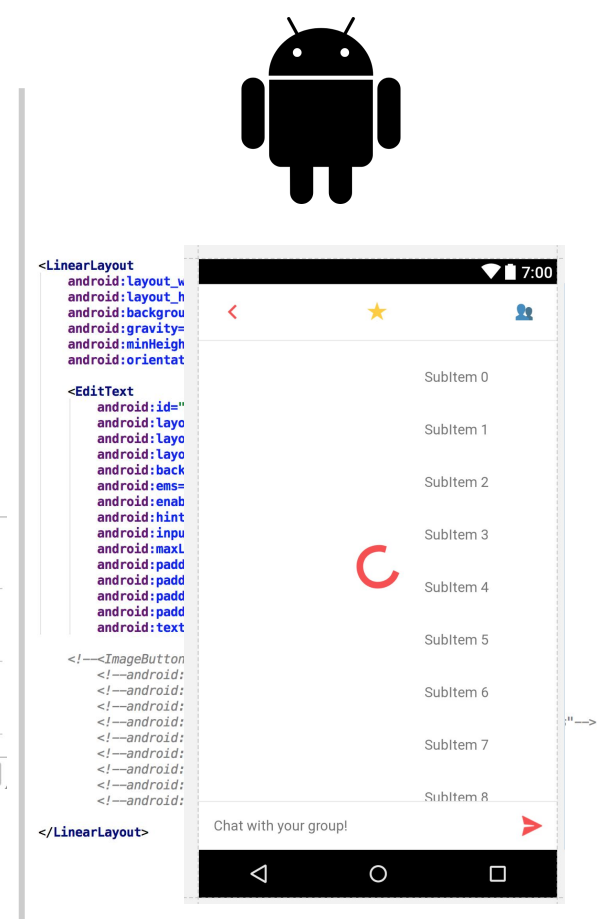
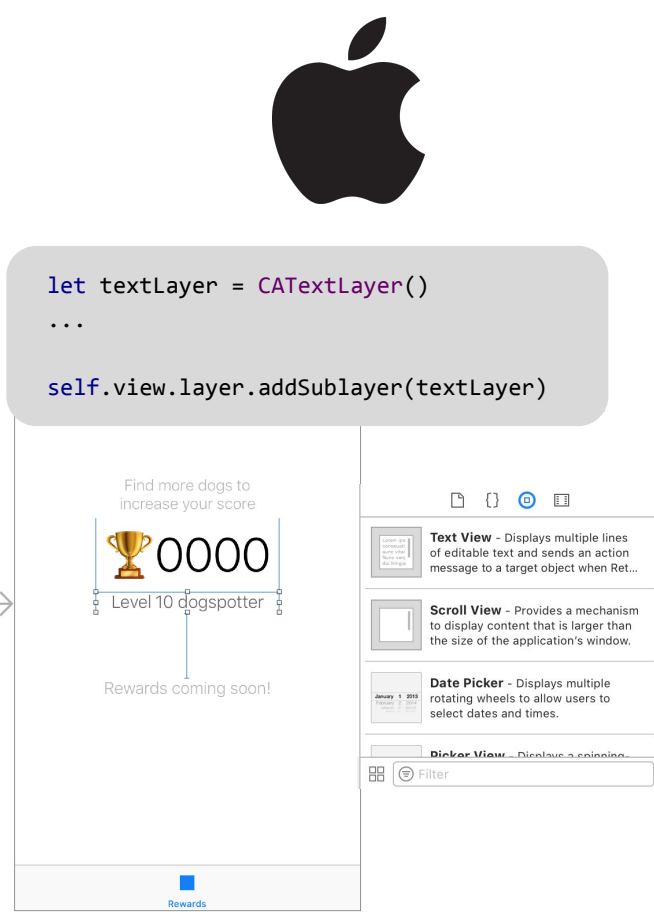


Modal — deco

Display content in a view which overlays the entire app.

```
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           Ctrl+R on the web for dev menu
27         </Text>
28       </View>
29     );
30   }
31 }
```

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

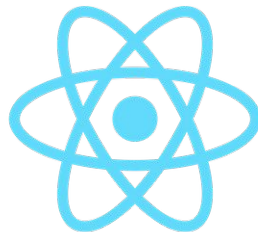



Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

Overview for today

- Introduce JavaScript Basics ✓
- Introduce JSX ✓
- Create an application from scratch
 - Breakdown React Native project files

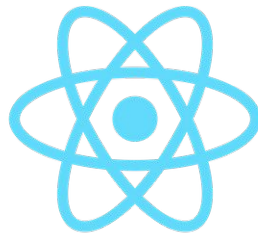


Last lecture

- Introduced CS47 and React Native as a framework
- Briefly touched upon environment setup
- Assignment 1 due Tuesday, October 1st, at 11:59 PM

Overview for today

- Introduce JavaScript Basics ✓
- Introduce JSX ✓
- ➔ Create an application from scratch
 - Breakdown React Native project files



Creating an empty project

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```

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

```
render() {  
  return ourNestedView;  
}
```

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

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```



```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```

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

```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```

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

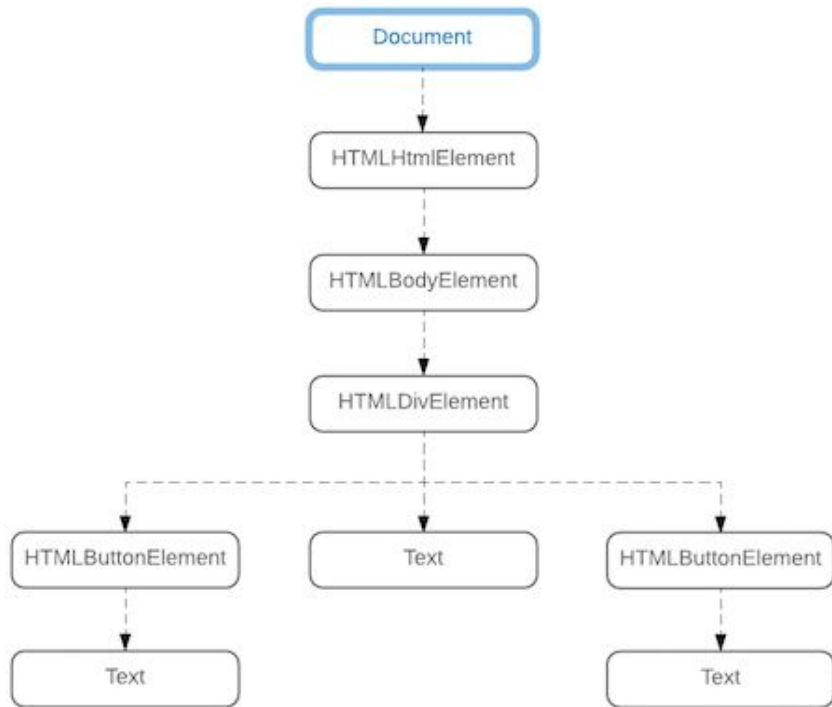
Sneak Peek: there's something called the "state" of a component. It's basically a variable that says what to render. When the values of this variable are changed, the render function is called again. You will learn more about a component's state on week 2.

Virtual DOM (Document Object Model)

A virtual representation of a desired view state

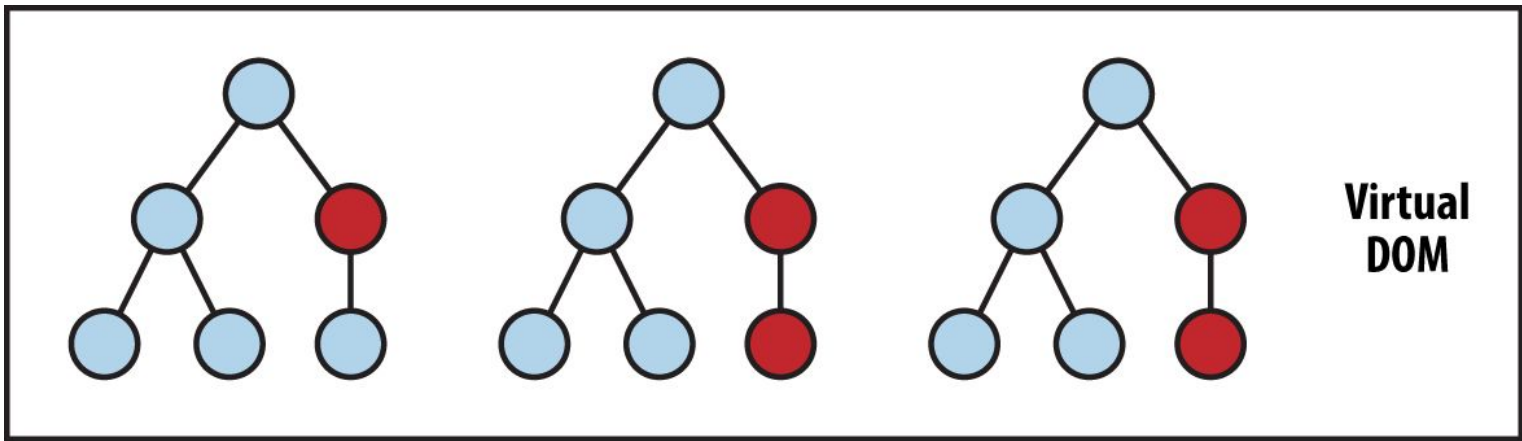
(not necessarily the same thing as the DOM on a browser)

Virtual DOM (Document Object Model)

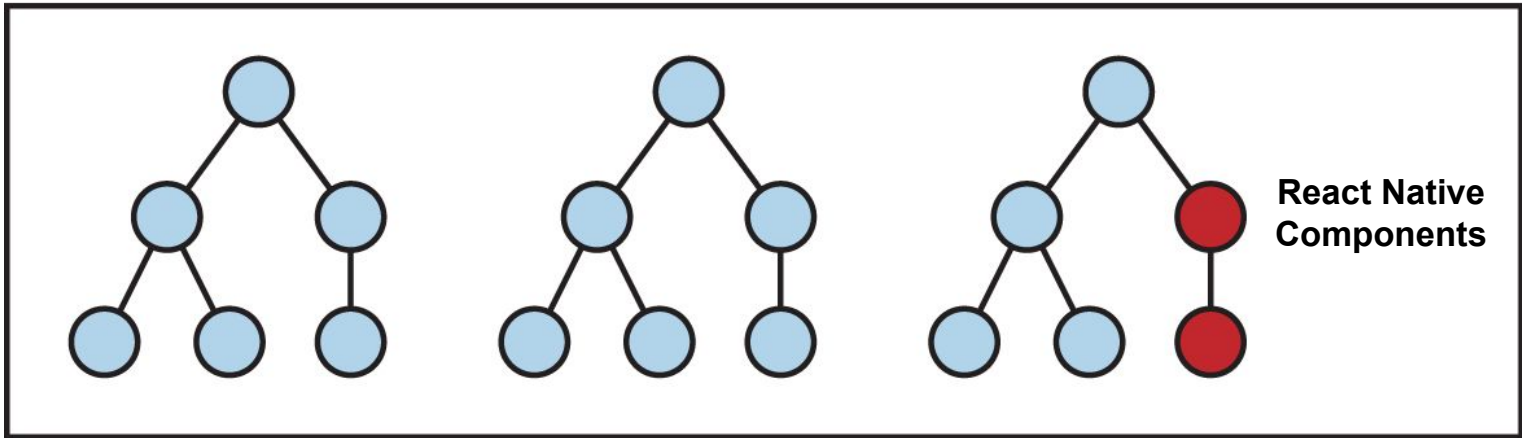


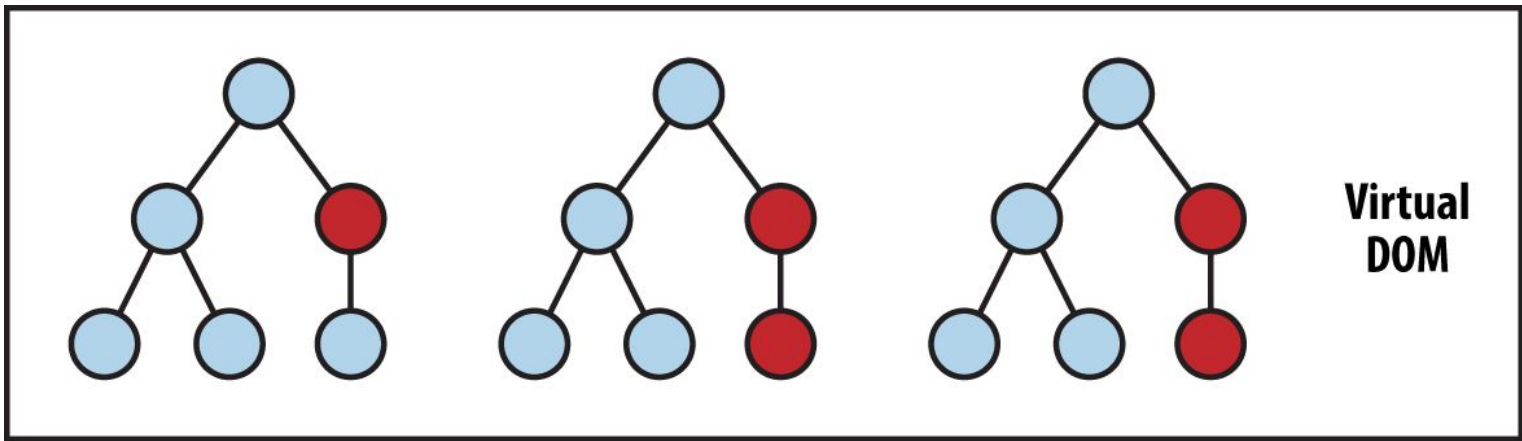
```
render() {  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <Text>Changes you make will automatically reload.</Text>  
      <Text>Shake your phone to open the developer menu.</Text>  
    </View>  
  );  
}
```

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

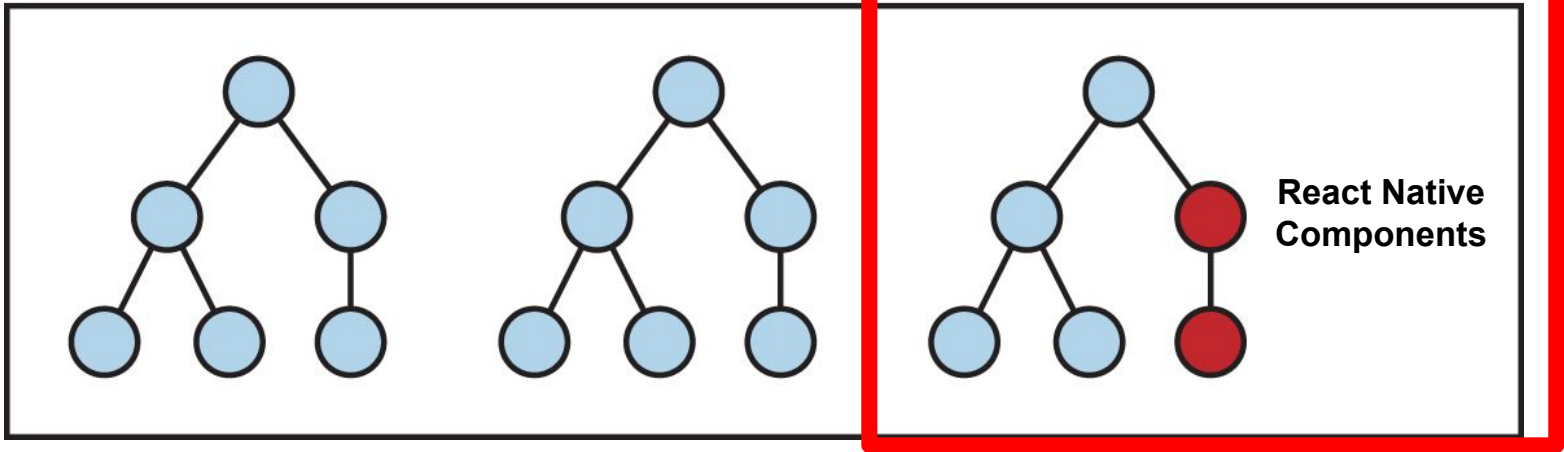


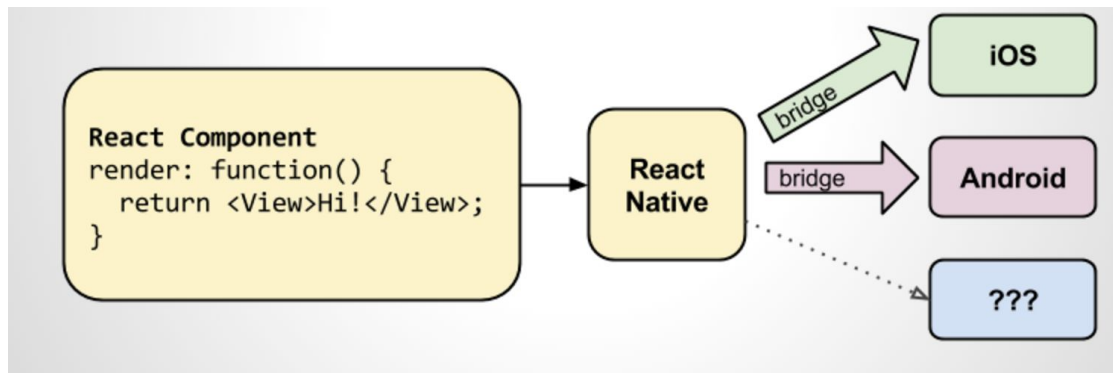
State Change → Compute Diff → Re-render





State Change → Compute Diff → Re-render






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



View



UIView



TextView



UILabel

[Back to project](#)

Project

▼ AwesomeProject

> .expo-shared

> .git

▼ assets

icon.png

splash.png

> node_modules

.gitignore

.watchmanconfig

App.js

app.json

babel.config.js

package-lock.json

package.json

yarn-error.log

.expo-shared | hidden folder, !important

.git | hidden folder, git info, !imp

assets | project assets e.g. icon

node_modules | all modules for project

.gitignore | hidden file, git stuff, !imp

.watchmanconfig | hidden file, !imp

App.js | Your code, very very important

app.json | app metadata, !too imp

Babel.config.js | !imp unless you know babel

Package-lock.json | auto gen, modules config

Package.json | packages of your project, imp

yarn-error.log | !imp, error log

```
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>

Office Hours

OH start on week 2. Email us directly if you're not available at these times

Abdallah AbuHashem

Monday (12-1 PM) @ Huang Basement

By appointment

Vy Mai

Tuesday (3-4 PM) @ Old Union

By appointment

Cisco Vlahakis

Wednesday (8-9 PM) @ Huang Basement

By appointment

Tiffany Manuel

Thursday (2-3 PM) @ Huang Basement

By appointment

CS47: Cross-Platform Mobile Development

Lecture 1B: Introduction to Javascript (ES6)

<https://cs-47.stanford.edu>



James Landay
Abdallah AbuHashem
Tiffany Manuel
Cisco Vlahakis
Vy Mai

Fall 2019