深入浅出 React Native：使用 JavaScript 构建原生应用

数月前，Facebook 对外宣布了正在开发的 React Native 框架，这个框架允许你使用 JavaScript 开发原生的 iOS 应用——就在今天，Beta 版的仓库释出了！

基于 PhoneGap 使用 JavaScript 和 HTML5 开发 iOS 应用已经有好几年了，那 React Native 有什么牛的？

React Native 真的很牛，让大家兴奋异常的主要原因有两点：

1.可以基于 React Native 使用 JavaScript 编写应用逻辑，UI 则可以保持全是原生的。这样的话就没有必要就 HTML5 的 UI 做出常见的妥协；

2.React 引入了一种与众不同的、略显激进但具备高可用性的方案来构建用户界面。长话短说，应用的 UI 简单通过一个基于应用目前状态的函数来表达。

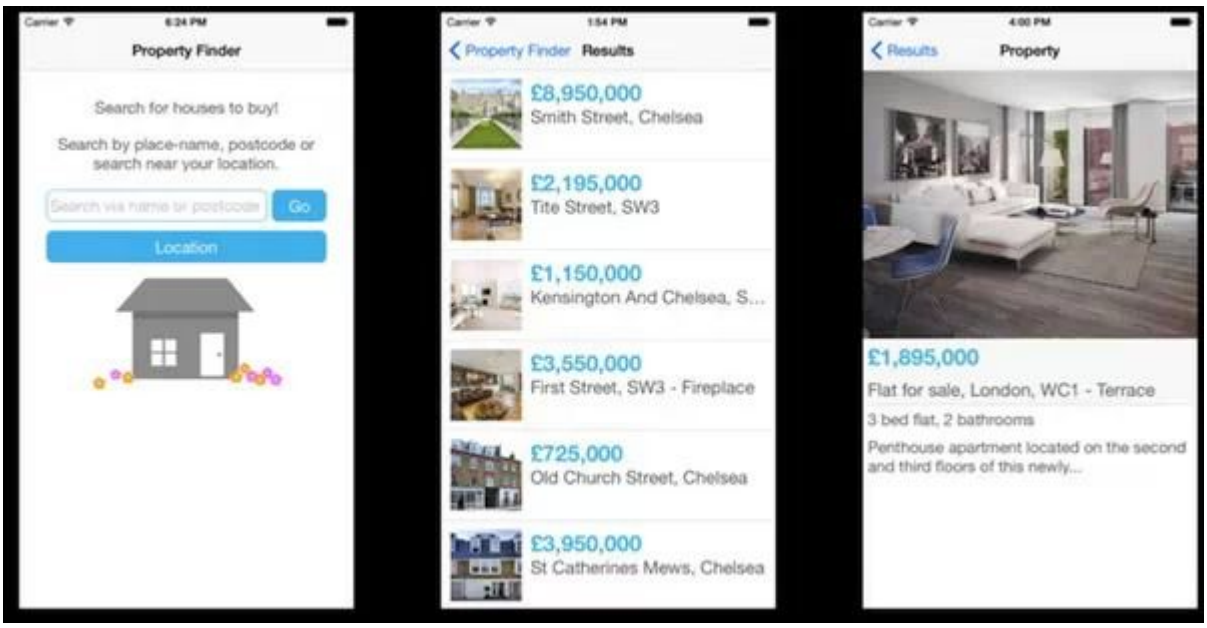
React Native 的关键就是，以把 React 编程模式的能力带到移动开发来作为主要目标。它的目标不是跨平台一次编写到处执行，而是一次学习跨平台开发。这个是一个非常大的区别。这篇教程只介绍 iOS 平台，不过你一旦掌握了相关的概念，就可以应用到 Android 平台，快速构建 Android 应用。

如果之前只用过 Objective-C 或者 Swift 写应用的话,你很可能不会对使用 JavaScript 来编写应用的愿景感到兴奋。尽管如此，作为一个 Swift 开发者来说，上面提到的第二点应该可以激起你的兴趣！

你通过 Swift，毫无疑问学习到了新的更多有效的编码方法和技巧，鼓励转换和不变性。然而，构建 UI 的方式还是和使用 Objective-C 的方式一致。仍然以 UIKit 为基础，独断专横。

通过像 virtual DOM 和 reconciliation 这些有趣的概念，React 将函数式编程直接带到了 UI 层。

这篇教程将带着你一路构建一个 UK 房产搜索应用：



如果你之前一点 JavaScript 都没写过，别担心。这篇教程带着你进行一步一步进行编码。React 使用 CSS 属性来定义样式，一般比较容易读也比较容易理解。但是如果你想了解更多的话，可以去看看 Mozilla Developer Network reference，很不错的。

想要学习更多，继续往下读！

准备工作

React Native 框架托管在 GitHub 上。你可以通过两种方式获取到它：使用 git 克隆仓库，或者下载一个 zip 压缩包文件。如果你的机器上已经安装了 React Native，在着手编码前还有其他几个因素需要考虑。

React Native 借助 Node.js，即 JavaScript 运行时来创建 JavaScript 代码。如果你已经安装了 Node.js，那就可以上手了。
首先，使用 Homebrew 官网提供的指引安装 Homebrew，然后在终端执行以下命令：

```
brew install node
```

接下来，使用 homebrew 安装 watchman，一个来自 Facebook 的观察程序：

```
brew install watchman
```

通过配置 watchman，React 实现了在代码发生变化时，完成相关的重建的功能。就像在使用 Xcode 时，每次保存文件都会进行一次创建。

React Native 有很多的依赖，需要在运行之前安装好。在 React Native 文件目录下打开一个终端，执行下面代码：

```
npm install
```

这里通过 Node 包管理器抓取到项目的所有依赖；功能上和 CocoaPods 或者 Carthage 类似。成功执行该命令后，你会发现一个 node_modules 文件夹被创建，包含了各种外部依赖。

最后，启动开发服务器。在刚才打开的终端中，执行下面命令：

```
npm start
```

执行上面命令，你会看到：


```
$ npm start
```

```
> react-native@0.1.0 start /Users/colineberhardt/Projects/react-native

> ./packager/packager.sh

| Running packager on port 8081.

| Keep this packager running while developing on any JS

| projects. Feel free to close this tab and run your own

| packager instance if you prefer.

| https://github.com/facebook/react-native
React packager ready.
```

就这样简单，准备开始！脚本在终端继续执行，我们继续。

至此，我推荐尝试一个 React Native 示例来测试配置项。在 react-native/Examples/Movies 文件夹下打开项目，然后创建并且运行它，确保你可以正确地发布这个 Movies 应用。

注意：在进入编码工作之前，还有最后一件事 —— 在这个教程中，你需要编写大量的 JavaScript 代码，Xcode 并非是最好的工具！我使用 Sublime Text，一个价格合理且应用广泛的编辑器。不过，atom，brackets 或者其他轻量的编辑器都能胜任这份工作。

React Native 你好

在开始“搜房 App”之前，先来个简单的 Hello World App 热热身。在这一节里，你将会使用到一些组件。

下载起始项目，解压缩到 react-native/Examples 目录中。解压完成后，在 Xcode 中打开 PropertyFinder 项目，不要直接运行这个项目，还需要加上一些 JS！

在编辑器中打开 PropertyFinderApp.js，将下面这行代码加到文件的开头位置：

```
'use strict';
```

这行代码是用于开启 Strict Mode，Strict mode 的错误处理可以有所提高，JavaScript 的一些语言缺陷也可以避免。简而言之就是，JavaScript 在这种模式下工作地更好！

注意：想要研究一下 Strict Mode 的朋友，我会推荐你阅读 Jon Resig 的文章:“ECMAScript 5 Strict Mode, JSON, and More”

然后，加上这一行：

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

这句代码是将 react-native 模块加载进来，并将它赋值给变量 React 的。React Native 使用同 Node.js 相同的模块加载方式：require，这个概念可以等同于 Swift 中的“链接库”或者“导入库”。

注意：想要了解更多关于 JavaScript 模块的知识，我推荐阅读 Addy Osmani 写的这篇文章。

在 require 语句的下面，加上这一段：

```
var styles = React.StyleSheet.create({
  text: {
    color: 'black',
    backgroundColor: 'white',
    fontSize: 30,
    margin: 80
  }
});
```

以上代码定义了一段应用在 “Hello World” 文本上的样式。如果你曾接触过 Web 开发，那你很可能已经发现了：React Native 使用的是 CSS 来定义应用界面的样式。

现在我们来关注应用本身吧！依然是在相同的文件下，将以下代码添加到样式代码的下面：

```
class PropertyFinderApp extends React.Component {
  render() {
    return React.createElement(React.Text, {style: styles.text}, "Hello World!");
  }
```

是的，就是 JavaScript class!

类 (class) 是在 ES6 中被引入的，纵然 JavaScript 一直在进步，但 Web 开发者受困于兼容浏览器的状况中，不能怎么使用 JS 的新特性。React Native 运行在 JavaScriptCore 中是，也就是说，你可以使用 JS 的新特性啦，完全不用担心兼容什么的呢。

注意：如果你是一名 Web 开发者，我百分百鼓励你要使用现代的 JavaScript，然后使用像 Babel 这样的工具生成兼容性的 JavaScript，用于支持兼容性不好的老浏览器。

PropertyFinderApp 继承了 React.Component（React UI 的基础模块）。组件包含着不可变的属性，可变的状态变量以及暴露给渲染用的方法。这会你做的应用比较简单，只用一个渲染方法就可以啦。

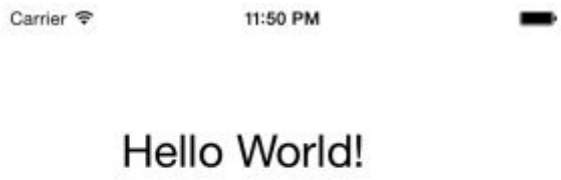
React Native 组件并不是 UIKit 类，它们只能说是在某种程度上等同。框架只是将 React 组件树转化成为原生的 UI。

最后一步啦，将这一行加在文件末尾：

```
React.AppRegistry.registerComponent('PropertyFinderApp', function() { return PropertyFinderApp });
```

AppRegistry 定义了 App 的入口，并提供了根组件。

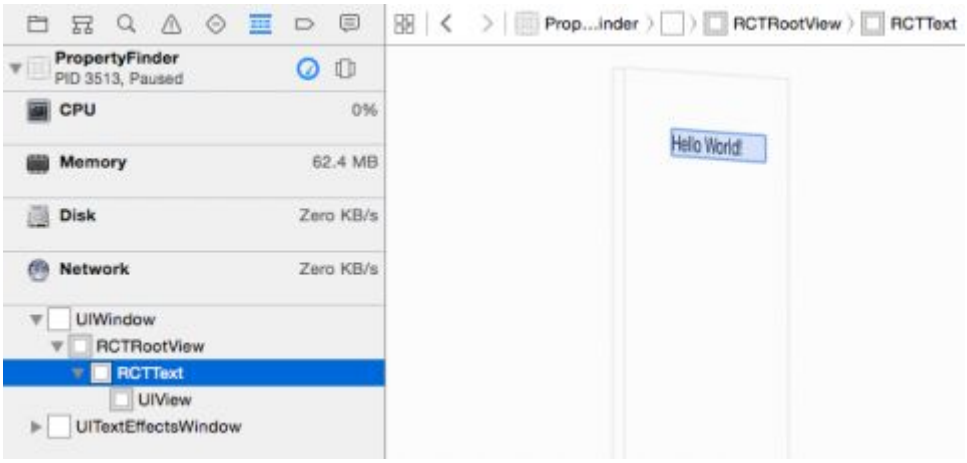
保存 PropertyFinderApp.js，回到 Xcode 中。确保 PropertyFinder 规划（scheme）已经勾选了，并设置了相应的 iPhone 模拟器，然后生成并运行你的项目。几秒之后，你应该就可以看到“Hello World” 应用正在运行了：



Hello World!

这个 JavaScript 应用运行在模拟器上，使用的是原生 UI，没有任何内嵌的浏览器哦！

还不相信这是真的？:] 那打开你的 Xcode，选择 Debug\View Debugging\Capture View Hierarchy，你看 native view hierarchy 中都没有 UIWebView，就只有一个原生的 view！:]



你一定很好奇其中的原理吧，那就在 Xcode 中打开 AppDelegate.m，接着找到 application:didFinishLaunchingWithOptions：这个方法构建了 RCTRootView 用于加载 JavaScript 应用以及渲染最后的视图的。

当应用开始运行的时候，RCTRootView 将会从以下的 URL 中加载应用：

```
http://localhost:8081/Examples/PropertyFinder/PropertyFinderApp.includeRequire.runModule.bundle
```

重新调用了你在运行这个 App 时打开的终端窗口,它开启了一个 packager 和 server 来处理上面的请求。

在 Safari 中打开那个 URL；你将会看到这个 App 的 JavaScript 代码。你也可以在 React Native 框架中找到你的 “Hello World” 代码。

当你的 App 开始运行了以后,这段代码将会被加载进来,然后 JavaScriptCore 框架将会执行它。在 Hello World 的例子里，它将会加载 PropertyFinderApp 组件，然后构建出原生的 UIKit 视图。关于这部分的内容，后文里会再详细解释的。

你好 JSX 的世界

你当前的应用程序会使用 React.createElement 来构建应用 UI ,React 会将其转换到原生环境中。在当前情况下，你的 JavaScript 代码是完全可读的,但一个更复杂的 UI 与嵌套的元素将迅速使代码变成一大坨。

确保应用程序仍在运行,然后回到你的文本编辑器中，编辑 PropertyFinderApp.js 。修改组件 render 方法的返回语句如下:

```
return <React.Text style={styles.text}>Hello World (Again)</React.Text>;
```

这是 JSX ,或 JavaScript 语法扩展,它直接在你的 JavaScript 代码中混合了类似 HTML 的语法;如果你是一个 web 开发人员,应该对此不陌生。在本篇文章中你将一直使用 JSX 。

把你的改动保存到 PropertyFinderApp.js 中，并返回到模拟器。按下 Cmd + R ,你将看到你的应用程序刷新,并显示更新的消息 “Hello World(again)”。

重新运行一个 React Native 应用程序像刷新 web 浏览器一样简单!:]

因为你会使用相同的一系列 JavaScript 文件,您可以让应用程序一直运行,只在更改和保存 PropertyFinderApp.js 后刷新即可

注意:如果你感到好奇,可以看看你的“包”在浏览器中，JSX 被转换成什么。

这个 “Hello World” 已经够大家玩耍了,是时候构建实际的应用程序了!

添加导航

我们的房产查找应用使用标准的栈式导航，基于 UIKit 的 navigation controller。现在正是添加的时候。

在 index.ios.js 文件中，把 PropertyFinderApp 重命名为 HelloWorld：

```
class HelloWorld extends React.Component {
```

“Hello World” 这几个字你还需要让它显示一会儿，但它不再是应用的根组件了。

接下来，在 HelloWorld 这个组件下面添加如下这个类：

```
class PropertyFinderApp extends React.Component {

render() {

return (


);

}

}
```

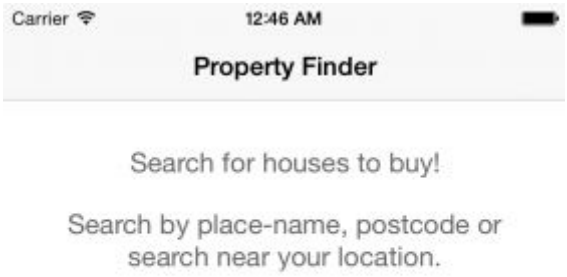
构造一个 navigation controller,应用一个样式,并把初始路由设为 Hello World 组件。在 Web 开发中，路由就是一种定义应用导航的一种技术，即定义页面——或者说是路由——与 URL 的对应关系。

在同一个文件中，更新样式定义，包含如下 container 的样式：

```
var styles = React.StyleSheet.create({
  text: {
    color: 'black',
    backgroundColor: 'white',
    fontSize: 30,
    margin: 80
  },
  container: {
    flex: 1
  }
});
```

在随后的教程中会告诉你 flex: 1 是什么意思。

回到模拟器，Cmd+R，看看新 UI 的样子：



这就是包含了 root view 的 navigation controller，目前 root view 就是 “Hello World”。很棒——应用已经有了基础的导航结构，到添加真实 UI 的时候了。

创建搜索页

在项目中添加一个新文件，命名为 SearchPage.js，然后将其放在 PropertyFinderApp.js 所在目录下。在文件中添加下面代码：

```
'use strict';
var React = require('react-native');
var {
  StyleSheet,
  Text,
  TextInput,
  View,
  TouchableHighlight,
  ActivityIndicatorIOS,
  Image,
  Component
} = React;
```

你会注意到，位于引入 react-native 所在位置的前面有一个严格模式标识，紧接着的声明语句是新知识。

这是一种解构赋值，准许你获取对象的多个属性并且使用一条语句将它们赋给多个变量。结果是，后面的代码中可以省略掉 React 前缀；比如，你可以直接引用 StyleSheet ，而不再需要 React.StyleSheet。解构同样适用于操作数组，更多细节请戳这里。

继续在 SearchPage.js 文件中添加下面的样式：

```
var styles = StyleSheet.create({
  description: {
    marginBottom: 20,
    fontSize: 18,
    textAlign: 'center',
    color: '#656565'
  },
  container: {
    padding: 30,
    marginTop: 65,
    alignItems: 'center'
  }
});
```

同样，以上都是标准的 CSS 属性。和 Interface Builder 相比，这样设置样式缺少了可视化，但是比起在 viewDidLoad() 中逐个设置视图属性的做法更友好！

只需要把组件添加到样式声明的前面：

```
class SearchPage extends Component {
  render() {
    return (
          Search for houses to buy!
          Search by place-name, postcode or search near your location.
    );
  }
}
```

render 很好地展示出 JSX 以及它表示的结构。通过这个样式，你可以轻易地描绘出组件 UI 的结构：一个容器，包含两个 text 标签。

最后，将下面的代码添加到文件末尾：

```
module.exports = SearchPage;
```

这可以 export SearchPage 类，方便在其他文件中使用它。

下一步是更新应用的路由，以初始化路由。

打开 PropertyFinderApp.js，在文件顶部紧接着上一个 require 语句的位置添加下面代码：

```
var SearchPage = require('./SearchPage');
```

在 PropertyFinderApp 类的 render 函数内部，通过更新 initialRoute 来引用最新添加的页面，如下：

```
component: SearchPage
```

此时，如果你愿意则可以移除 HelloWorld 类以及与它相关联的样式。你不在需要那段代码了。

切换到模拟器，按下 Cmd+R 查看新的 UI：

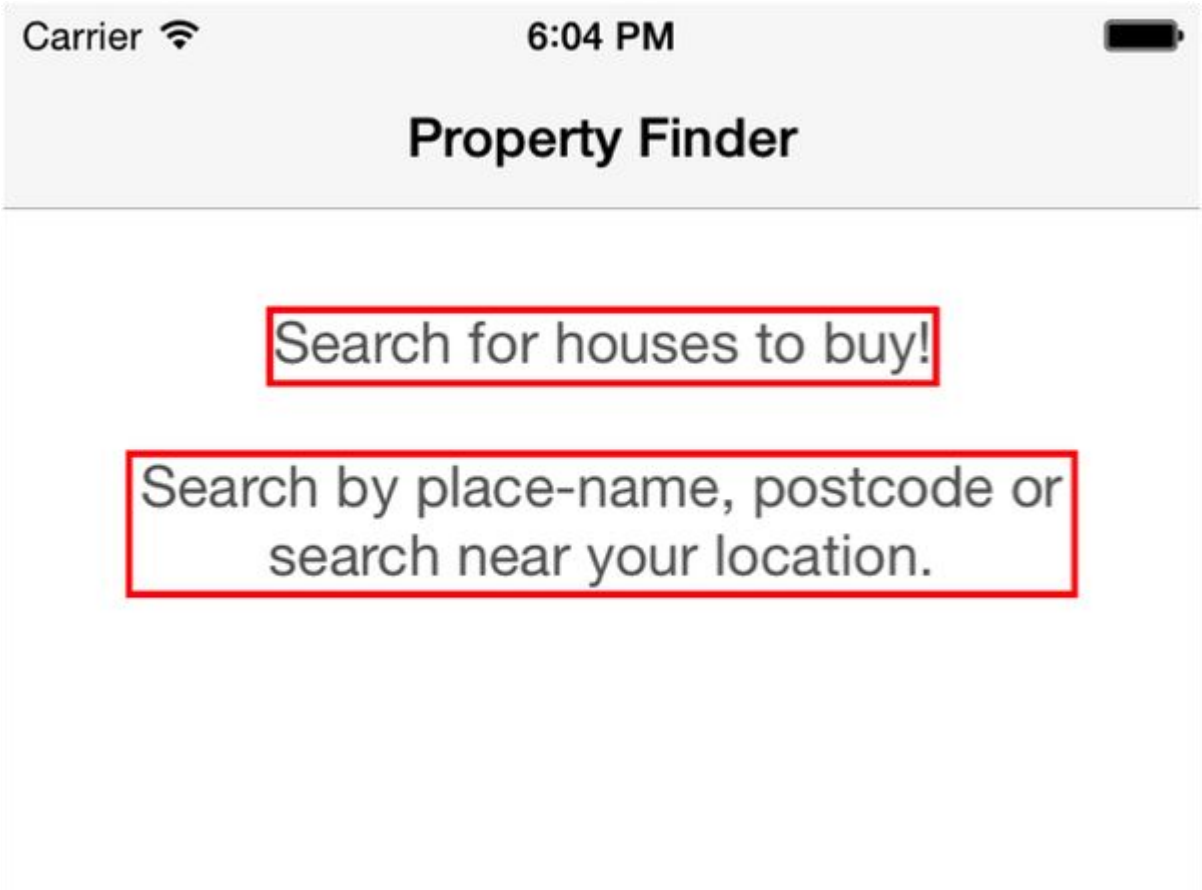
使用 Flexbox 定义外观

现在，你已经看到了用基本的 CSS 属性来控制外间距（margin），内间距（padding）还有颜色（color）。不过，可能你还不太了解要如何使用伸缩盒（flexbox），flexbox 是最近新加入 CSS 规范，用它就能很便利地布局界面。

React Native 用 css-layout（这是一个用 JavaScript 实现 flexbox 标准然后编译成 C（iOS 平台）或者 Java（Android 平台）的库)。

Facebook 把这个项目单独出来实在太正确了，这样可以编译成多种语言，促进更多新颖的应用的发展，比如 flexbox layout to SVG。

在你的 App 中，容器（container）默认地是纵向布局，也就是说在它的子元素将会竖直地排列，像这样：



这被称为主轴 (main axis)，它的方向可以是竖直的也可以是水平的。

每一个子元素在竖直方向上的位置是由它的 margin，height 和 padding 共同决定的。容器的 alignItems 属性也要设置成 center，这个属性可以控制子元素在十字轴上的位置。在这里，它实现了居中对齐的文本。

好啦，现在我们把输入框和按钮加上去吧。打开 SearchPage.js，将下面的代码插入第二个 Text 元素的后面：

```
<View style={styles.flowRight}>
```

```
  <TextInput
  style={styles.searchInput}
  placeholder='Search via name or postcode'/>
  <TouchableHighlight style={styles.button}
  underlayColor='#99d9f4'>
  <Text style={styles.buttonText}>Go</Text>
  </TouchableHighlight>
  </View>
  <TouchableHighlight style={styles.button}
  underlayColor='#99d9f4'>
  <Text style={styles.buttonText}>Location</Text>
  </TouchableHighlight>
```

现在你已经加上了两个最高等级的视图（top-level view），一个视图包含了文本输入框和一个按钮，还有一个视图内只有一个按钮。在后文中你会看到，它们的样式是什么样的。

接着，添加上对应的样式：

```
flowRight: {
  flexDirection: 'row',
  alignItems: 'center',
  alignSelf: 'stretch'
},
buttonText: {
  fontSize: 18,
  color: 'white',
  alignSelf: 'center'
},
button: {
  height: 36,
  flex: 1,
  flexDirection: 'row',
```

```
  backgroundColor: '#48BBEC',
  borderColor: '#48BBEC',
  borderWidth: 1,
  borderRadius: 8,
  marginBottom: 10,
  alignSelf: 'stretch',
  justifyContent: 'center'
},
searchInput: {
  height: 36,
  padding: 4,
  marginRight: 5,
  flex: 4,
  fontSize: 18,
  borderWidth: 1,
  borderColor: '#48BBEC',
  borderRadius: 8,
  color: '#48BBEC'
}
```

要注意格式问题：每一个样式都是用逗号分隔开的，所以别忘了在 container 选择器后面还要加上一个逗号。

以上的样式将会应用在你刚刚加上的输入框和按钮上。

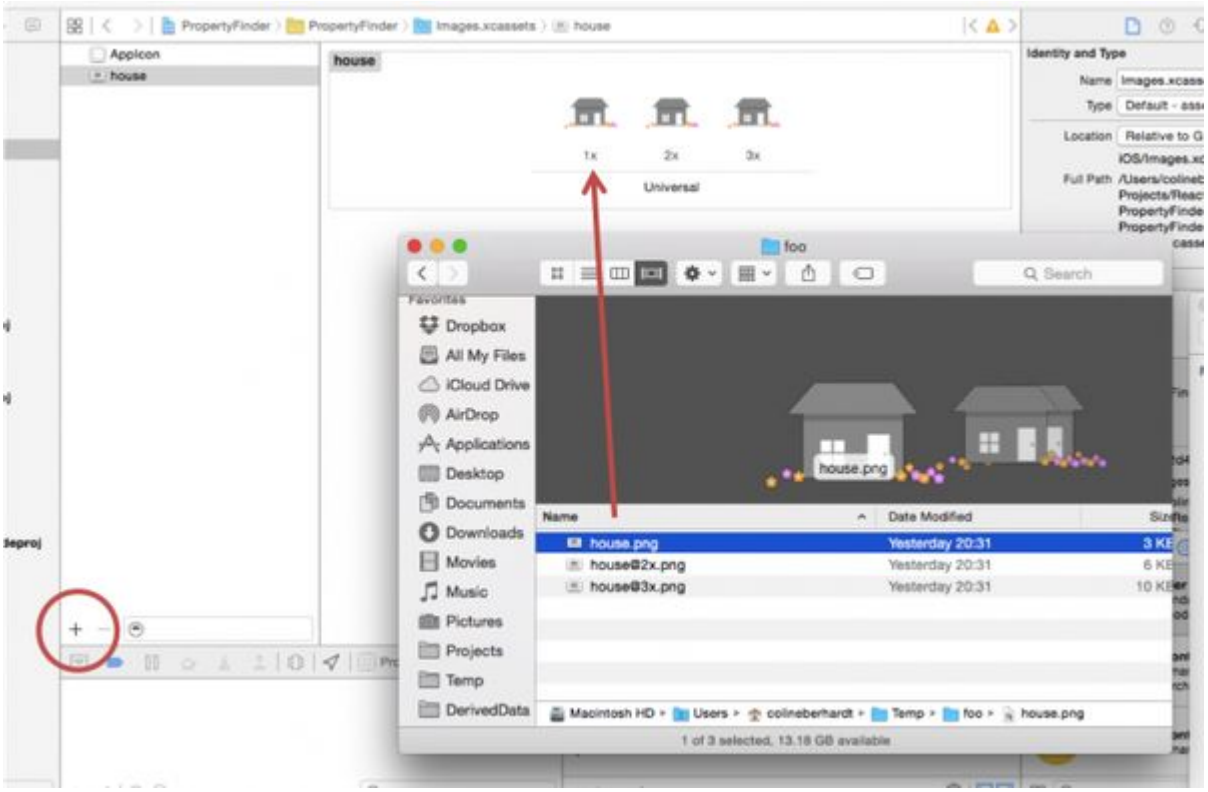
现在返回到模拟器，然后按下 Cmd+R 刷新界面：

文本区域和 ’Go’ 按钮在同一行，不需要显式地定义两个组件的宽度，你只需要将它们放在同一个容器中，加上 flexDirection:’row’ 样式，再定义好它们的 flex 值。文本区域是 flex:4，按钮则是 flex:1，这说明两者的宽度比是 4:1。

大概你也发现了，你的“按钮”其实并不是按钮！:] 使用了 UIKit 后，按钮更倾向于是可以轻碰（tap）的标签（label），所以 React Native 团队决定直接在 JavaScript 中构建按钮了。所以你在 App 中使用的按钮是 TouchableHighlight，这是一个 React Native 组件，当轻碰 TouchableHighlight 时，它会变得透明从而显示出衬底的颜色（也就是按钮下层的组件颜色）。

搜索界面的最后一步就是加上一张图。你可以从这里下载我们用的图片素材并解压。

在 Xcode 中打开 Images.xcassets 文件，点击加号添加一个新的图片集。然后将图片素材拖进正确的“区间”：



你需要重启应用才能让图片生效。

将以下代码添加到 TouchableHighlight 组件后面，它将用于“获取位置”按钮：

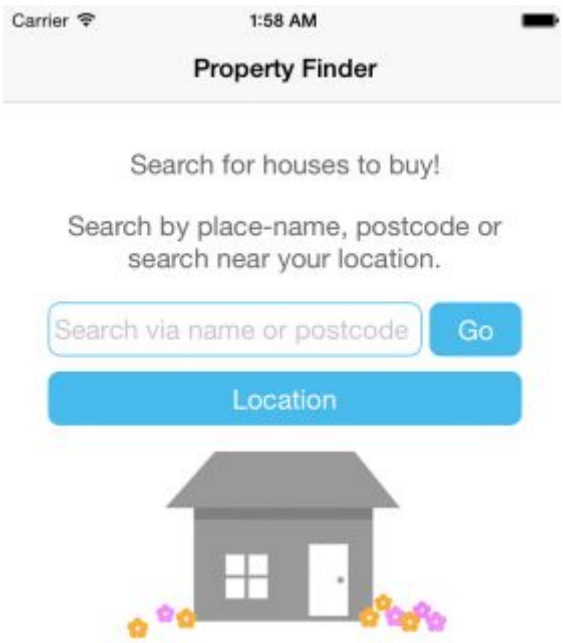
```
<Image source={require('image!house')} style={styles.image}/>
```

现在再样式表的最后加上图片对应的样式，别忘了给原样式中最后一个加上逗号哦：

```
image: {

width: 217,

height: 138

}
```

require('image!house') 语句用于确定在你应用的 asset 目录下的图片资源，在 Xcode 中，如果你的打开了 Images.xcassets，你会看到一个“房屋”的图标，正是上面代码中引用到的。

返回到模拟器，Cmd+R 刷新 UI：



注意：如果你这会没有看到“房屋”图片，取而代之的是一张“找不到资源”的图片，尝试重启 packager（也就是在终端里输入 npm start 命令）。