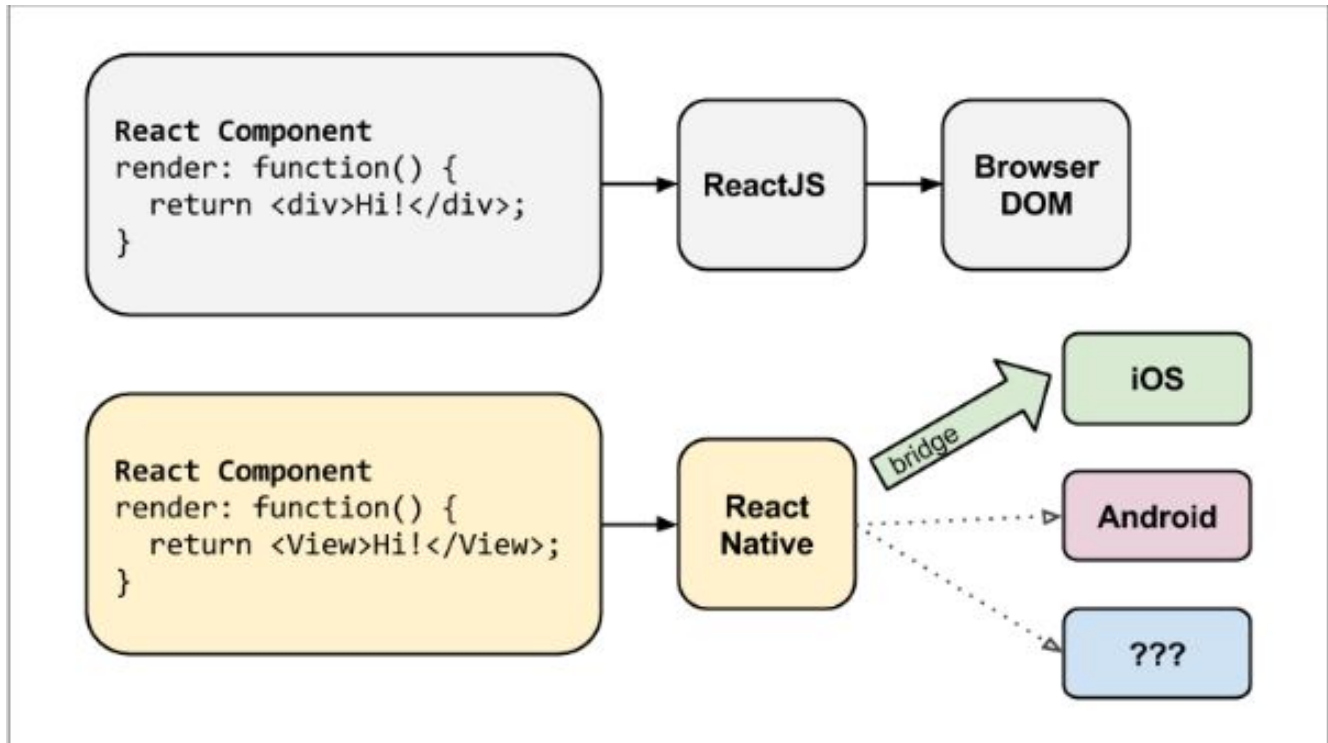


- **1. Introduce**

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components

How Does React Native Work?



The bridge translates APIs and UI elements from their host platform underpinnings into interfaces accessible via Javascript

- React native writing mobile applications in Javascript it use a mechanisms call Virtual DOM

What is Virtual DOM?

-Virtual DOM acts as a layer between the developer's description of how things ought to look, and the work done to actually render them onto the UI.

-It is a drawing, contains all information necessary to create a DOM, you can create, delete, update on it but do not need a direct impact into the DOM.

-For more detail about Virtual DOM and DOM please visit:

<https://tonyfreed.blog/what-is-virtual-dom-c0ec6d6a925c#.ucp70ff91>

<https://viblo.asia/nghiamt/posts/ZWApGx7eM06y>

Why use React Native?

Learn once, write anywhere:

-Build on IOS and Android platform

-Rather than focusing on code reuse, React Native prioritizes knowledge reuse. It allows you to carry your knowledge and skillset across platforms.

Leveraging the native platform

React Native gives you the benefits of native development, and allows you to take advantage of whatever platform you are developing for. This includes UI elements and animations, as well as platform-specific APIs, such as geolocation or the camera roll.

We can contrast this with existing cross-platform mobile solutions. Many of them allow you to develop inside of thinly-disguised browser components, which then re-implement pieces of the native UI. However, these applications often feel a little “off.” It is usually impossible to fully replicate native UI elements, and as a result, aspects of your application such as animations and styling can feel clunky or just less-than-natural. React Native lets you neatly circumvent this problem.

Compared with traditional mobile application development, React Native does not have quite the same access to its host platform. There is a slight layer of added complexity to using APIs not yet supported by the React Native core, for instance, and synchronous APIs in particular pose a challenge.

Using existing platform knowledge

For developers already accustomed to writing traditional native applications on a given platform, moving to React Native does not obsolete your platform-specific knowledge. Far from it; there are a number of circumstances in which this knowledge becomes critical. Occasionally, you will discover that the React Native “bridge” for a given platform does not yet support host platform APIs or features that you will need. For instance, being comfortable working with Objective-C enables you to jump in and add to the React Native “bridge” when necessary, thereby extending React Native’s support for iOS.

Community support will likely play a large factor in how React Native evolves on new platforms. Developers who are able to contribute back to the platform by building out the bridge will be better equipped to explore using React Native on new platforms, as well as lay the groundwork for its success

Risks and Drawbacks

The largest risk that comes with using React Native is related to its maturity level. As a young project, React Native will inevitably contain its share of bugs and unoptimized implementations. This risk is somewhat offset by its community. Facebook, after all, is using React Native in production, and the community contributions have been proceeding at a good tempo.

Similarly, by using React Native, you are relying heavily on a library that is not necessarily endorsed by the host platform you are working on. Because the Native team may not be able to coordinate with Apple, for instance, we can imagine that new iOS releases could require a scramble to get the React Native library up to speed. For most platforms, however, this should be a relatively minor concern.

The other major drawback is what happens when you encounter a limitation in React Native. Say that you want to add support for a host platform API not yet built into the React Native core library. If you are a developer who is only comfortable in JavaScript, there is a nontrivial learning curve to add this support yourself..

- **2. Getting Started**

Setting Up Environment (<http://facebook.github.io/react-native>)

Node, Watchman:

You will need to use Homebrew, a common package manager for OS X, in order to install React Native's dependencies. From the command line:

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

Note: [Watchman](#) is a tool by Facebook for watching changes in the filesystem. It is highly recommended you install it for better performance.

The React Native CLI:

Node.js comes with npm, which lets you install the React Native command line interface. Run the following command in a Terminal:

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

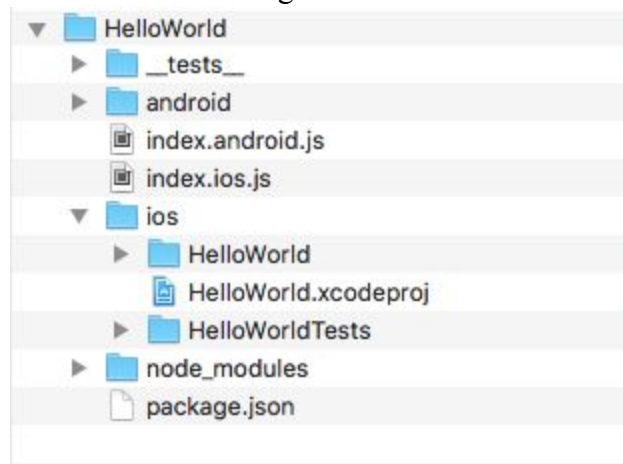
Explore Sample app

Create Sample app:

You can use the React Native command line tools to create a new application. This will generate a fresh project with all of the React Native and iOS boilerplate for you

```
react-native init SampleProject
```

The resulting directory should have the following structure:



Application Structure

All test file in `_test_` directory

All iOS boilerplate code is located in the iOS directory.

React code is located in the `index.ios.js` file, which is in the project root.

Node modules, as usual, can be found in the `node_modules` folder.

Package.json is Pod file in iOS

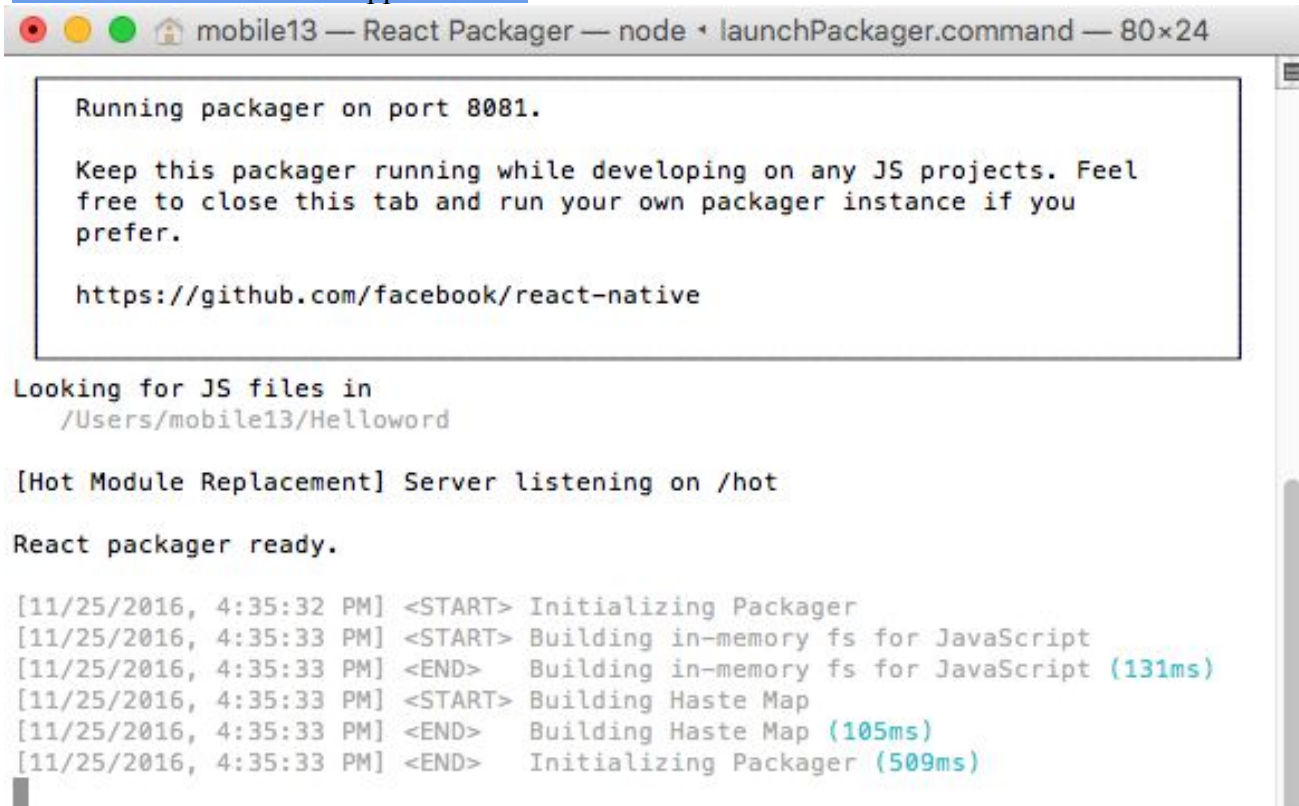
Running app in simulator:

```
cd SampleProject
```

react-native run-ios

You also use Xcode and open file .project in IOS folder and running it with XCode

Now you can see terminal as follows, This is allowed application running in background, don't turn off it. You can use Xcode build app on device



```
mobile13 — React Packager — node • launchPackager.command — 80x24

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

Looking for JS files in
/Users/mobile13/Helloworld

[Hot Module Replacement] Server listening on /hot

React packager ready.

[11/25/2016, 4:35:32 PM] <START> Initializing Packager
[11/25/2016, 4:35:33 PM] <START> Building in-memory fs for JavaScript
[11/25/2016, 4:35:33 PM] <END> Building in-memory fs for JavaScript (131ms)
[11/25/2016, 4:35:33 PM] <START> Building Haste Map
[11/25/2016, 4:35:33 PM] <END> Building Haste Map (105ms)
[11/25/2016, 4:35:33 PM] <END> Initializing Packager (509ms)
```

Attaching a component to the view

Open AppDelegate.m file and attention to:

```
RCTRootView *rootView = [[RCTRootView alloc] initWithBundleURL:jsCodeLocation
                                                    moduleName:@"HelloWorld"
                                                    initialProperties:nil
                                                    launchOptions:launchOptions];
```

The React Native library prefixes all of its classes with RCT, **RCTRootView** is a React Native class, in this case it is a root React view.

The **moduleName** passed to the RCTRootView determines which component will be mounted in the view. This is where you can choose which component should be rendered by your application

In order to use the HelloWorld component here, you need to register a React component with the same name. If you open up index.ios.js, you'll see that this is accomplished on the last line:

```
AppRegistry.registerComponent('HelloWorld', () => HelloWorld);
```

Index.ios.js

In this file you can see React Native Import some Component which was used in this file
Every class in React Native will render one View.

Developer tools

-Do not need to rebuild application in order to see your changes reflected; instead, you can hit CMD+R

to refresh your application just as you would any other webpage

-You will need to rebuild your app for changes to take effect in certain situations:

- You have added new resources to your native app's bundle, such as an image in Images.xcassets on iOS or the res/drawable folder on Android.
- You have modified native code (Objective-C/Swift on iOS or Java/C++ on Android)

-Use whatever text editor you prefer for Javascript editing : TextEditor, SublimeText, Atom, Deco ..

Using the Chrome Debugger

<https://facebook.github.io/react-native/docs/debugging.html#chrome-developer-tools>

Checking with Flow

Flow is a Javascript library for static type-checking. It relies on type inference to detect type errors even in unannotated code, and allows you to slowly add type annotations to existing projects

Install Flow with terminal: *brew install flow*

Checking with terminal: *flow check*

```

HelloWorld — -bash — 80x24
[Trungs-Mac:HelloWorld trungquangnguyen$ flow check
Skipping /Users/trungquangnguyen/Desktop/React-native/HelloWorld/flow/: No such
file or directory

node_modules/react-native/Libraries/Animated/src/AnimatedImplementation.js:1066
1066:   return this._interpolation(parentValue);
      ^^^^^^ AnimatedInterpolation. This type is incompatible with
235:   const val = +interpolations[i++](input);
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ array type. See: node_modules/react-nativ
e/Libraries/Animated/src/Interpolation.js:235

node_modules/react-native/Libraries/BatchedBridge/MessageQueue.js:275
275:   callback.apply(null, args);
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ call of method `apply`. Function cannot be c
alled on possibly null value
275:   callback.apply(null, args);
      ^^^^^^^^^ null

node_modules/react-native/Libraries/BatchedBridge/MessageQueue.js:275
275:   callback.apply(null, args);
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ call of method `apply`. Function cannot be c
alled on possibly undefined value
275:   callback.apply(null, args);
      ^^^^^^^^^ undefined
```

You can see many errors related to files in *node_modules* , you may need to add this line to your *flowconfig* under *[ignore]* : *./node_modules/.**

Flow check again and this is result:

```

[Trungs-Mac:HelloWorld trungquangnguyen$ flow check
Skipping /Users/trungquangnguyen/Desktop/React-native/HelloWorld/flow/: No such
file or directory

index.ios.js:13
13: } from 'react-native';
    ^^^^^^^^^^^^^^^^^^^^^^ react-native. Required module not found

Found 1 error
Trungs-Mac:HelloWorld trungquangnguyen$
```

Testing with Jest:

<https://facebook.github.io/jest/>

- React Native supports testing of React components using Jest. Jest is a unit testing framework built on top of Jasmine. It provides aggressive automocking of dependencies, and it meshes nicely with React's testing utilities.
- Jest will recursively search for files in a tests directory, and run them.
- Install Jest: *npm install jest-cli --save-dev*
- Run Jest: *npm test*

Building a Weather App Demo