Hi.

Bob Scarano

Organizing Code for Cross-Platform Web and Mobile Development

Bob Scarano

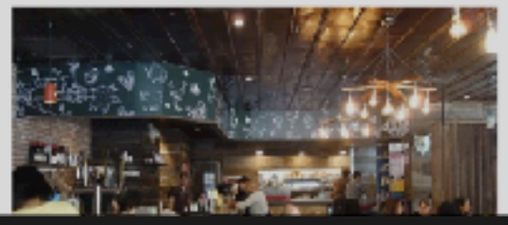
~~Organizing Code for Cross-Platform Web and Mobile Development~~

How we do Cross-Platform


HAIN REACT

SAVE CANCEL Editing 'Home'

LOCALLY SOURCED • CRAFTED WITH LOVE



Located in Park Slope, Pacific Tavern is a purveyor of fine

PHONE

(347) 555-1234

LOCATION

Dashboard

DEATH BY XOKO

Introducing

Small Batch Chocolate

Button Label

Style | Email

Button

Link /home

Style Solid

Alignment Center

Padding

COLORS

Text

Button

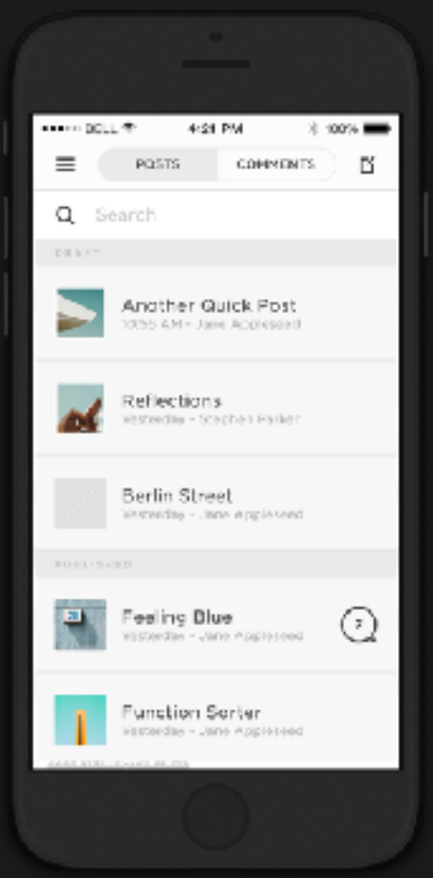
Background

GLOBAL STYLES

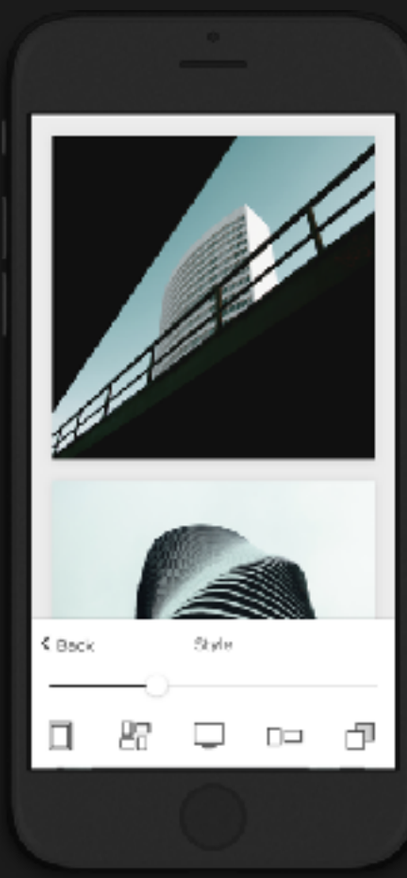
Font Pack Narrow Sans

Delete

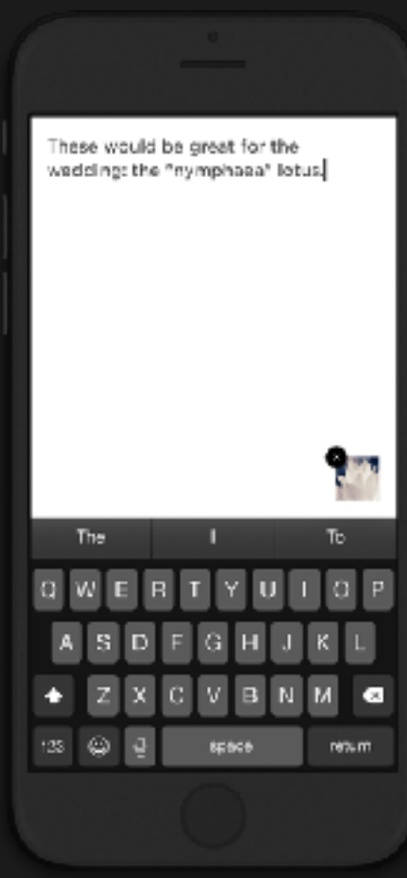
Blog



Portfolio

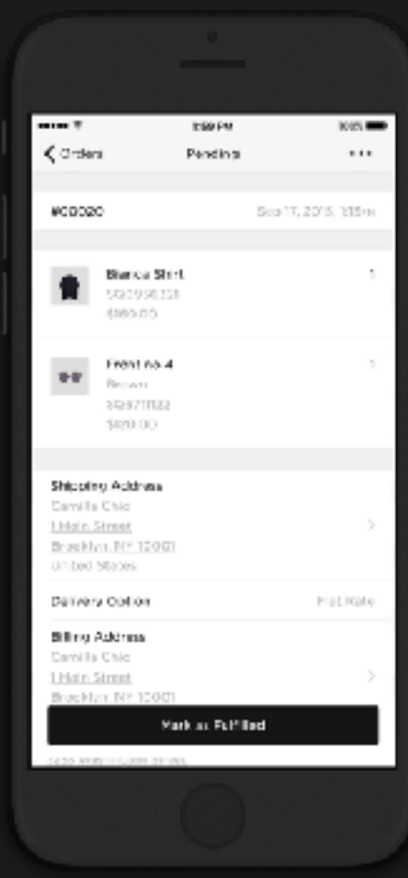


Note

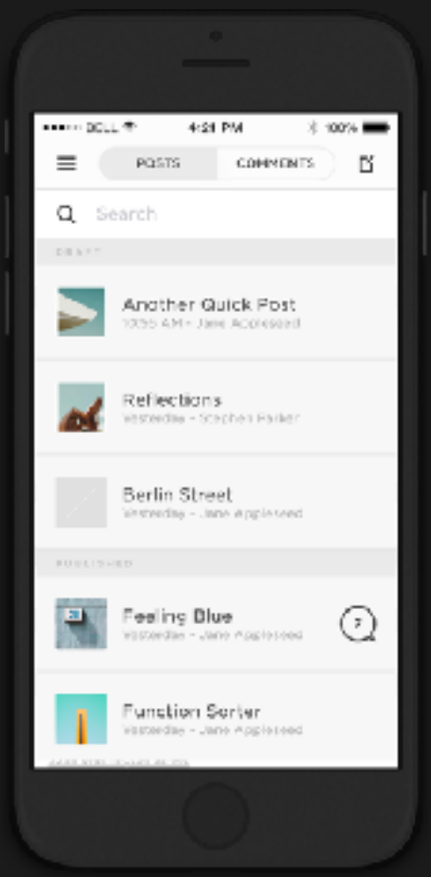


Analytics

Commerce

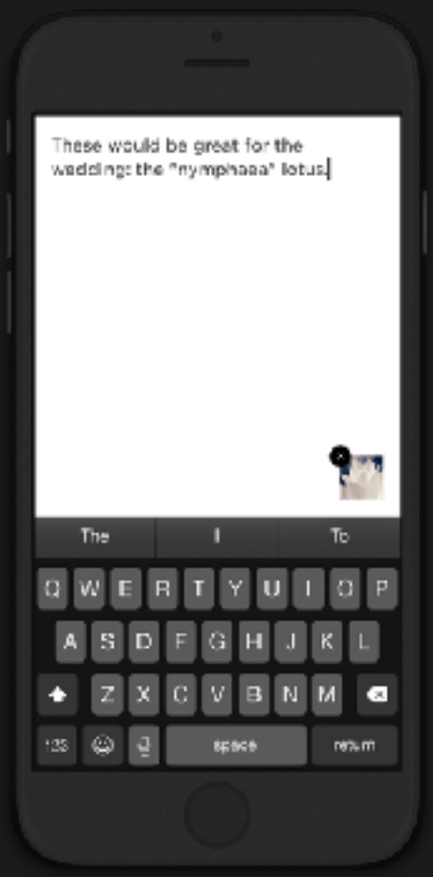


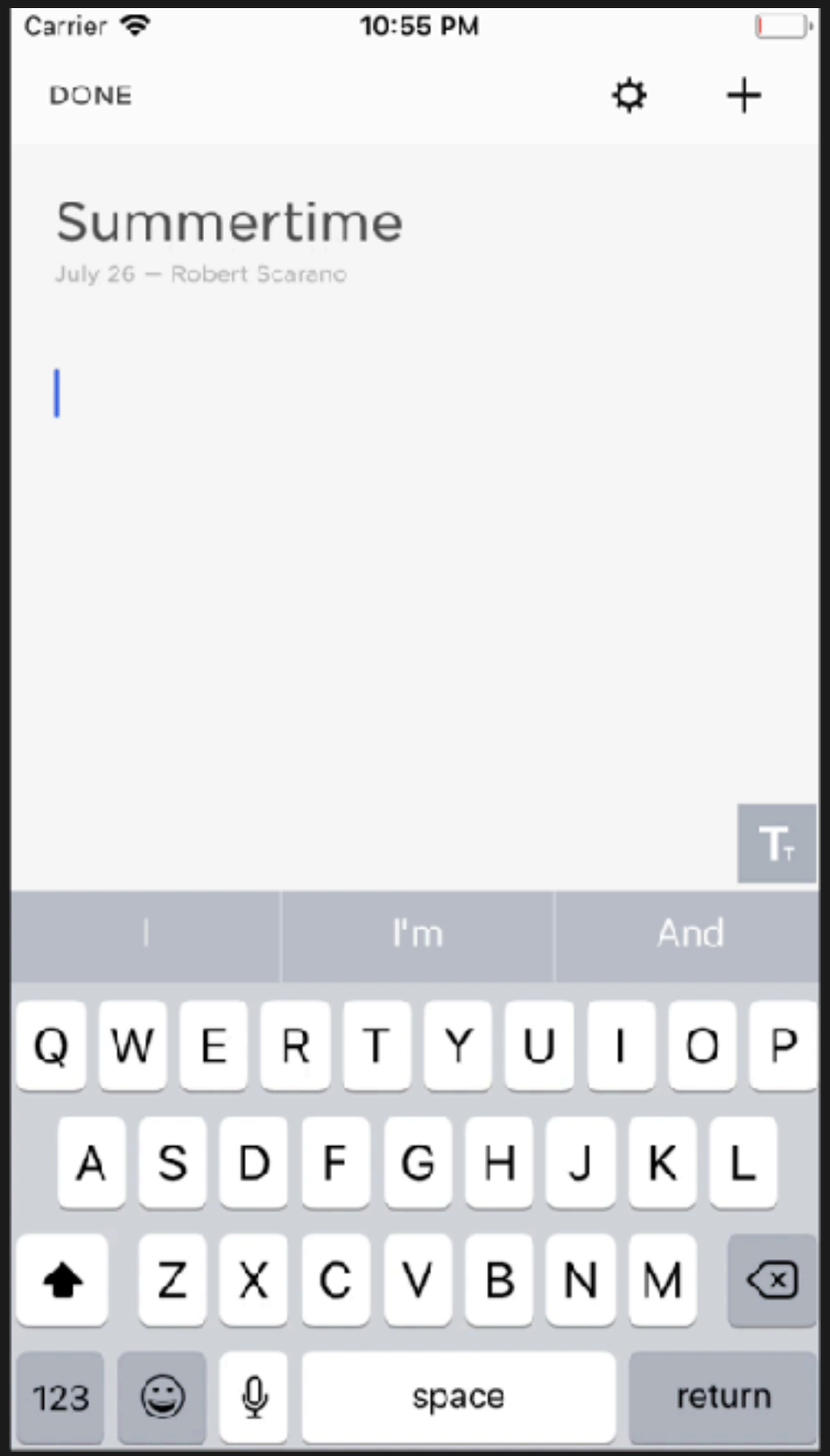
Blog



Portfolio

Note




Carrier
10:55 PM
DONE
Summertime
July 26 — Robert Scarano
I
I'm
And
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

Architecture

Build a Component Native Integration

~~Write it 3x~~

```
$ react-native init AwesomeProject
$ npm install
```

/node_modules

React

React Native

Dependencies

Native App

AppDelegate

ViewController

/node_modules

React

React Native

Dependencies

React-Native Components

App.js

Components

Native App

AppDelegate

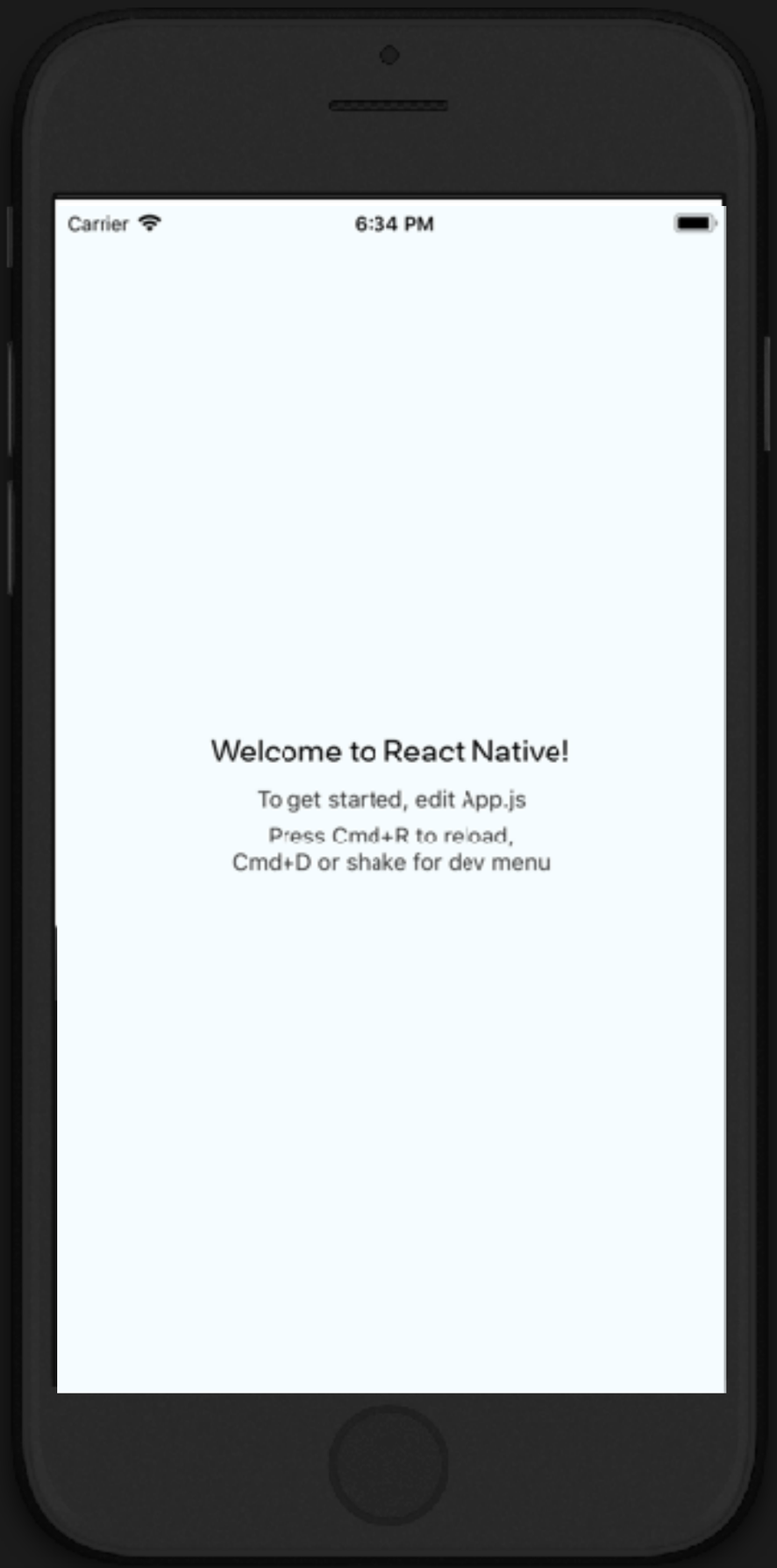
ViewController

/node_modules

React

React Native

Dependencies


Carrier
6:34 PM
Welcome to React Native!
To get started, edit App.js
Press Cmd+R to reload,
Cmd+D or shake for dev menu

```
$ create-react-app AwesomeProject
$ npm install
```

/node_modules

React

Dependencies

Web App

App.js

Components

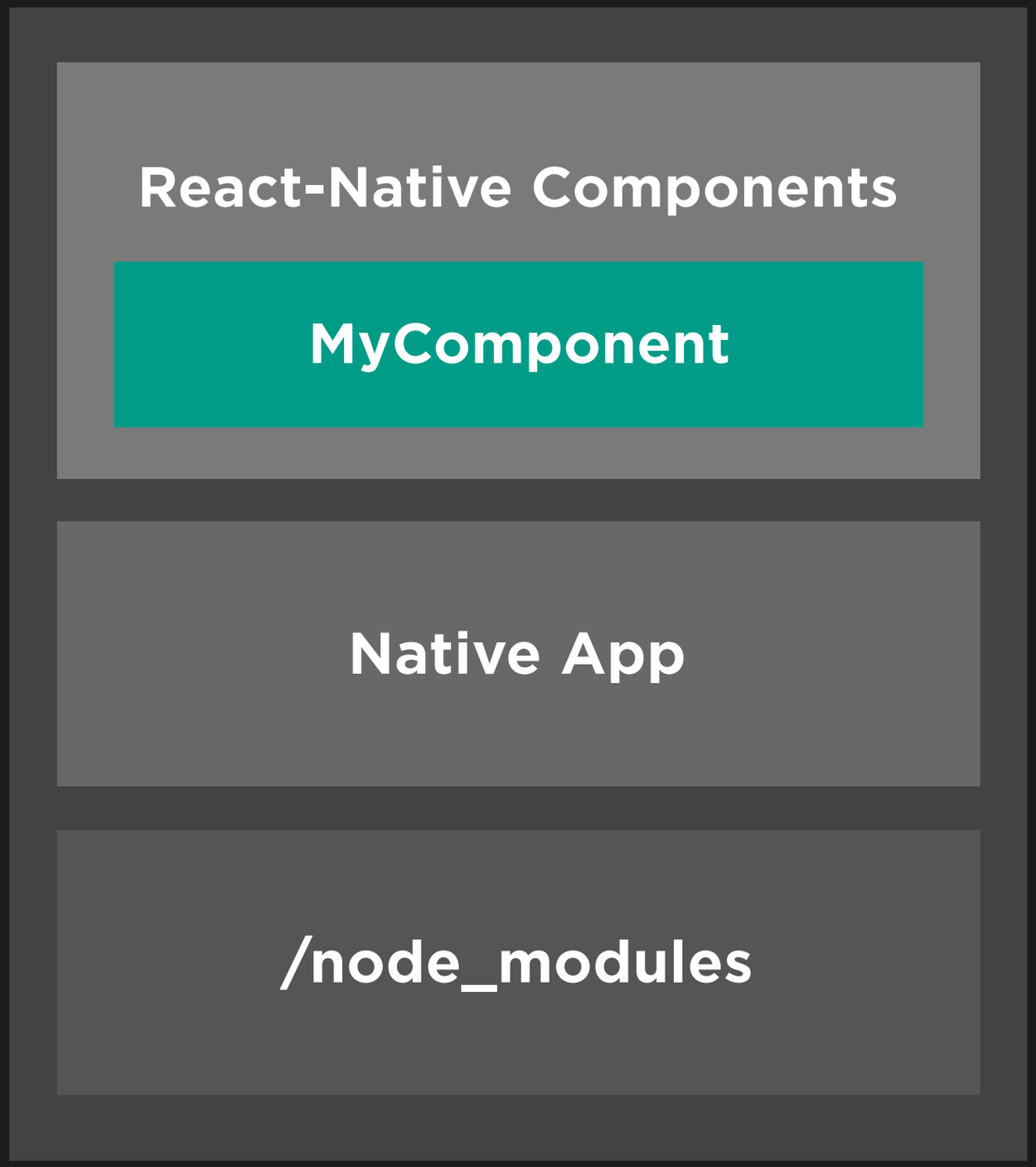
/node_modules

React

Dependencies

Welcome to React

To get started, edit `src/App.js` and save to reload.



React-Native App

React-Native Components

MyComponent

Native App

/node_modules

React-Native App

Web App

MyComponent

/node_modules

Web App

React-Native Components

MyComponent

Native App

/node_modules

React-Native App

Web App

MyComponent

/node_modules

Web App

Cross-Platform Components

MyComponent

Native App

/node_modules

/node_modules

React-Native App

Web App

Cross-Platform Components

MyComponent

Native App

/node_modules

react-native-web
react-native-primitives

/node_modules

React-Native App

Web App

~~Write it 1x~~

Shared

MyComponent

React-Native App

Web App

React-Native Components

MyComponent

Web App

MyComponent

Shared

MyComponent

React-Native App

Web App

Dashboard

DEATH BY XOKO

Introducing

Small Batch Chocolate

...

Button Label

Style | Email

Back | Link

/bananas-are-great

SECONDARY NAVIGATION

No collections in this navigation.

MAIN NAVIGATION

- Blog
- Let's Jam
- HELLO
- New Page ✓
- Sideways
- Monster Gallery
- MyFolderPage
- Calibration
- My folder name

Open in a new window

SECONDARY NAVIGATION

No collections in this navigation.

MAIN NAVIGATION

Blog

Let's Jam

HELLO

New Page

Sideways

Monster Gallery

MyFolderPage

Write it 1.5x

30% Web/Mobile

70% Shared Code

Code Organization

```
module/
  reactnative/
shared/
web/
package.json
```

```
module/
   reactnative/
   shared/
   MyComponent.js
   index.js
web/
```

```
module/
   reactnative/
      index.js
      MyComponent.js
      package.json
shared/
web/
   index.js
   MyComponent.js
   package.json
```

```
['module-resolver', {
  'alias': {
    'target': ‘./react-native’
  }
}]
```

```
['module-resolver', {
  'alias': {
    'target': ‘./web'
  }
}]
```

```
import Thing from ‘target/MyModule’
```

./shared/TelnumInput.js

```
class TelnumInput extends Component
```

./shared/TelnumInput.js

```
import numberLoader from
  'target/TelnumInput/numberLoader';

class TelnumInput extends Component
```

./shared/TelnumInput.js

```
import numberLoader from
  'target/TelnumInput/numberLoader';

class TelnumInput extends Component {

async componentWillMount() {
  const { asYouType } = await
    numberLoader();
  this.formatter = new asYouType();
}
```

./shared/TelnumInput.js

```
import numberLoader from
  'target/TelnumInput/numberLoader';

class TelnumInput extends Component {

format(value) {
  formattedValue =
    this.formatter.input(value);
  return formattedValue;
}
```

./react-native/TelnumInput.js

```
import BaseTelnumInput from
  'shared/TelnumInput';
import StringInput from
  'target/StringInput';

class TelnumInput extends Component
```

./react-native/TelnumInput.js

```
class TelnumInput extends Component {
  render() {
    const formattedValue =
      this.format(this.props.value);

    return (
      <StringInput
        autoCapitalize="none"
        autoCorrect={false}
        keyboardType="phone-pad"
        label={this.props.label}
        placeholder={this.props.placeholder}
        value={formattedValue}
      />
    );
  }
}
```

./react-native/TelnumInput.js

```
class TelnumInput extends Component {
  render() {
    const formattedValue =
      this.format(this.props.value);

    return (
      <StringInput
        autoCapitalize="none"
        autoCorrect={false}
        keyboardType="phone-pad"
        label={this.props.label}
        placeholder={this.props.placeholder}
        value={formattedValue}
      />
    );
  }
```

./react-native/TelnumInput.js

```
class TelnumInput extends Component {
  render() {
    const formattedValue =
      this.format(this.props.value);

    return (
      <StringInput
        autoCapitalize="none"
        autoCorrect={false}
        keyboardType="phone-pad"
        label={this.props.label}
        placeholder={this.props.placeholder}
        value={formattedValue}
      />
    );
}
```


TelephoneField US formatting (default

./web/TelnumInput.js

```
import BaseTelnumInput from
  'shared/TelnumInput';
import StringInput from
  'target/StringInput';

class TelnumInput extends Component
```

./web/TelnumInput.js

```
class TelnumInput extends Component {
  render() {
   const formattedValue =
      this.format(this.props.value);

    return (
      <StringField
        {...props}
        className={className}
        htmlAttributes={htmlAttributes}
        type="tel"
        label={label}
        placeholder={placeholder}
       value={formattedValue}
      />
    );
}
```

./web/TelnumInput.js

```
class TelnumInput extends Component {
  render() {
    const formattedValue =
      this.format(this.props.value);

    return (
      <StringField
        {...props}
        className={className}
        htmlAttributes={htmlAttributes}
        type="tel"
        label={label}
        placeholder={placeholder}
        value={formattedValue}
      />
    );
}
```

US formatting (default)

</>

This is the basic usage of the 'TelephoneField' field

undo

Component-Module

Component

Component

Component

AppDelegate

Component-Module

Component

Component

Native App

React-Native

AppDelegate

Native App

App Components

Component

Component

Component-Module

Component

Component

React-Native

AppDelegate

Application Classes

Networking

Persistence

Libraries

Native App

App Components

Component

Component

Component-Module

Component

Component

React-Native

AppDelegate

Component

Networking

Persistence

Libraries

Native App

App Components

Component

Component

Component-Module

Component

Component

React-Native

Component

Component

Native App

React-Native

Wrapper ViewController

Native

React Native

Wrapper ViewController

Dependencies

Component

Component

Native

React Native

Wrapper ViewController

Application Components

App Components

Dependencies

Component

Component

Native

React Native

Wrapper ViewController

AppDelegate

Application Components

App Components

Dependencies

Component

Component

Native

React Native

Native App

Wrapper ViewController

AppDelegate

Native Classes

Native App

AppDelegate

Native Classes

App.js

package.json

node_modules

Native App

AppDelegate

Native Classes

App.js

package.json

node_modules

Component

Component

Native App

App Components

AppDelegate

Native Classes

App.js

package.json

node_modules

Native App

ReactBridge

ReactViewController

ObjectReference

RN Toolkit

Component

Component

App Components

```
app/
  react/
    node_modules/
      MyComponent/
        Android/
        iOS/
          MyComponentViewController.swift
        index.js
        MyComponent.js
```

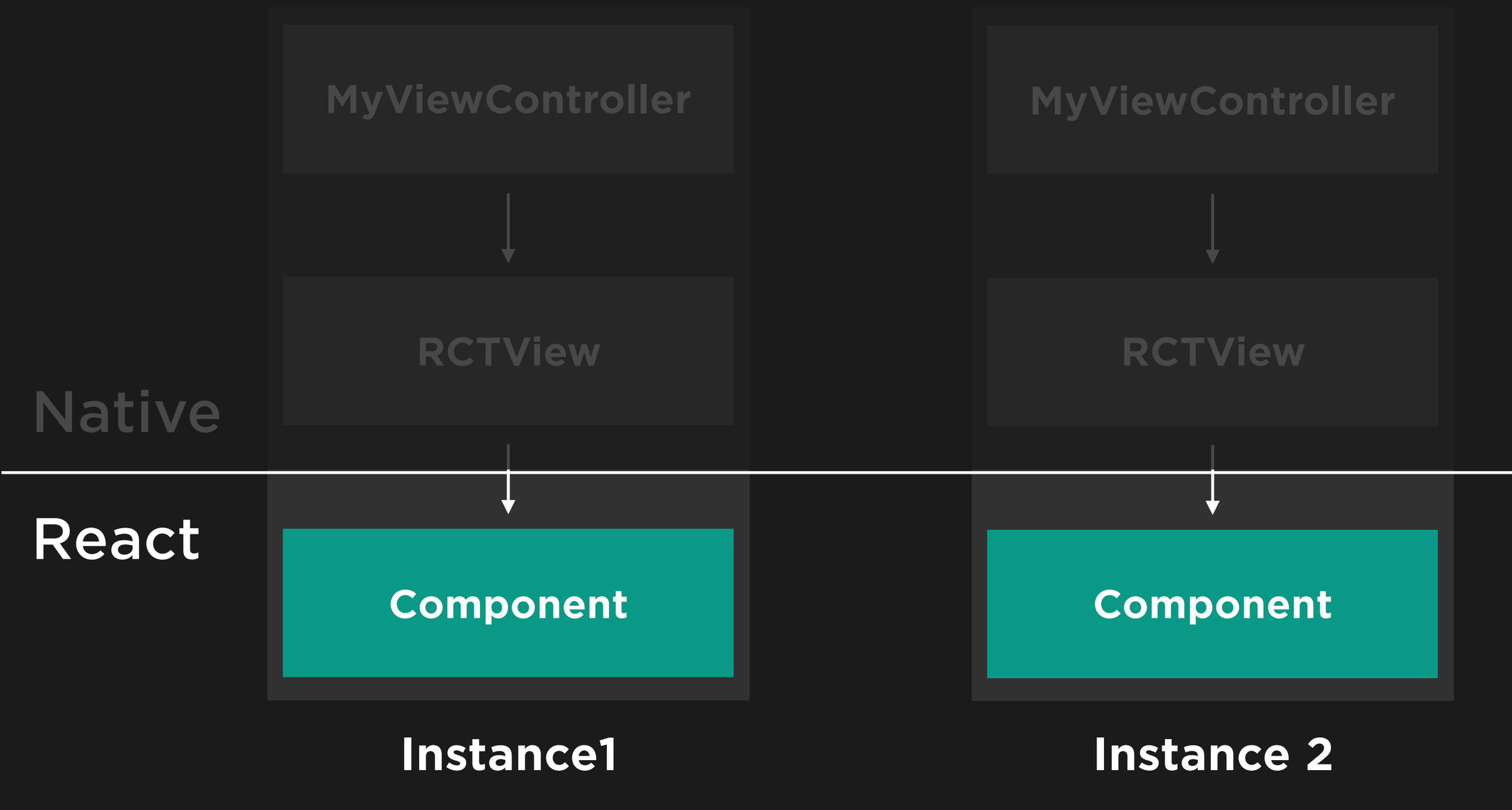
```
app/
  react/
    node_modules/
      MyComponent/
        Android/
        iOS/
          MyComponentViewController.swift
        index.js
        MyComponent.js
```

```
app/
  react/
    node_modules/
      MyComponent/
        Android/
        iOS/
          MyComponentViewController.swift
        index.js
        MyComponent.js
```

```
public class MyComponentViewController:
  ReactViewController {

    override open var componentName: String {
      return "MyReactComponent"
    }

    override open func props() -> [String: AnyObject]
      return [“title”: “Hello World”]
    }
```

```
public class MyComponentViewController:
  ReactViewController {

    override open var componentName: String {
        return "MyReactComponent"
    }

    override open func props() -> [String: AnyObject]
        return [“title”: “Hello World”]
    }
```

```
public class ReactBridge {
    public static let shared: RCTBridge?
}
```


MyViewController
RCTView
Native
React
Component
Instance1
MyViewController
RCTView
Component
Instance 2

MyViewController

RCTView

Component

MyViewController

RCTView

Component

Native

React

Instance1

Instance 2

NativeModuleManager

MyViewController

RCTView

Component

MyViewController

RCTView

Component

Native

React

Instance1

Instance 2

Native Module Documentation

Native modules are Objective-C classes that are available in JS.

Typically one instance of each module is created per JS bridge.

The fact that native modules are singletons limits the mechanism in context of embedding.

That said, we would need to keep a mapping from identifiers to native views in the module.

NativeModuleManager

MyViewController

MyViewController

RCTView

RCTView

Native

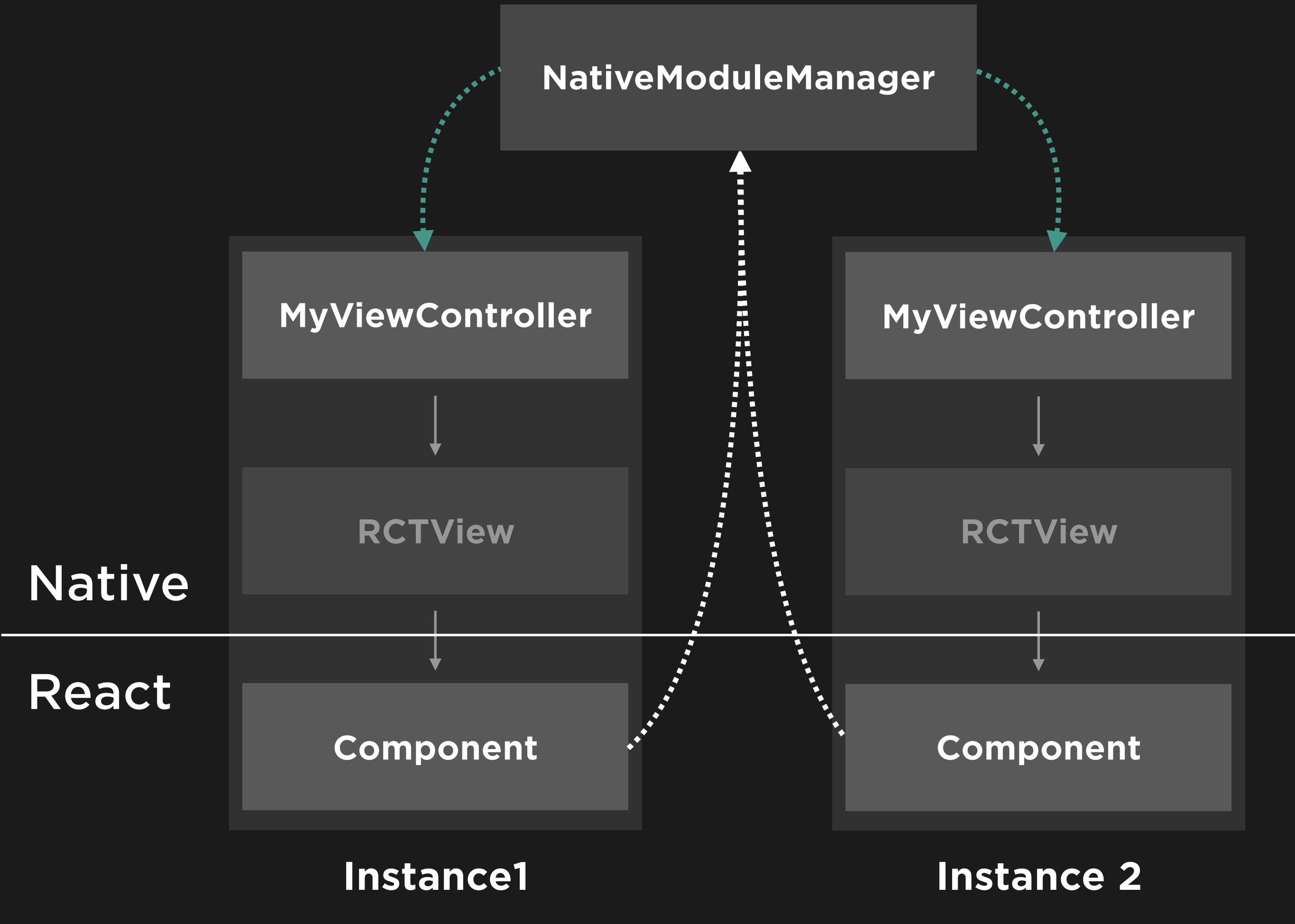
React

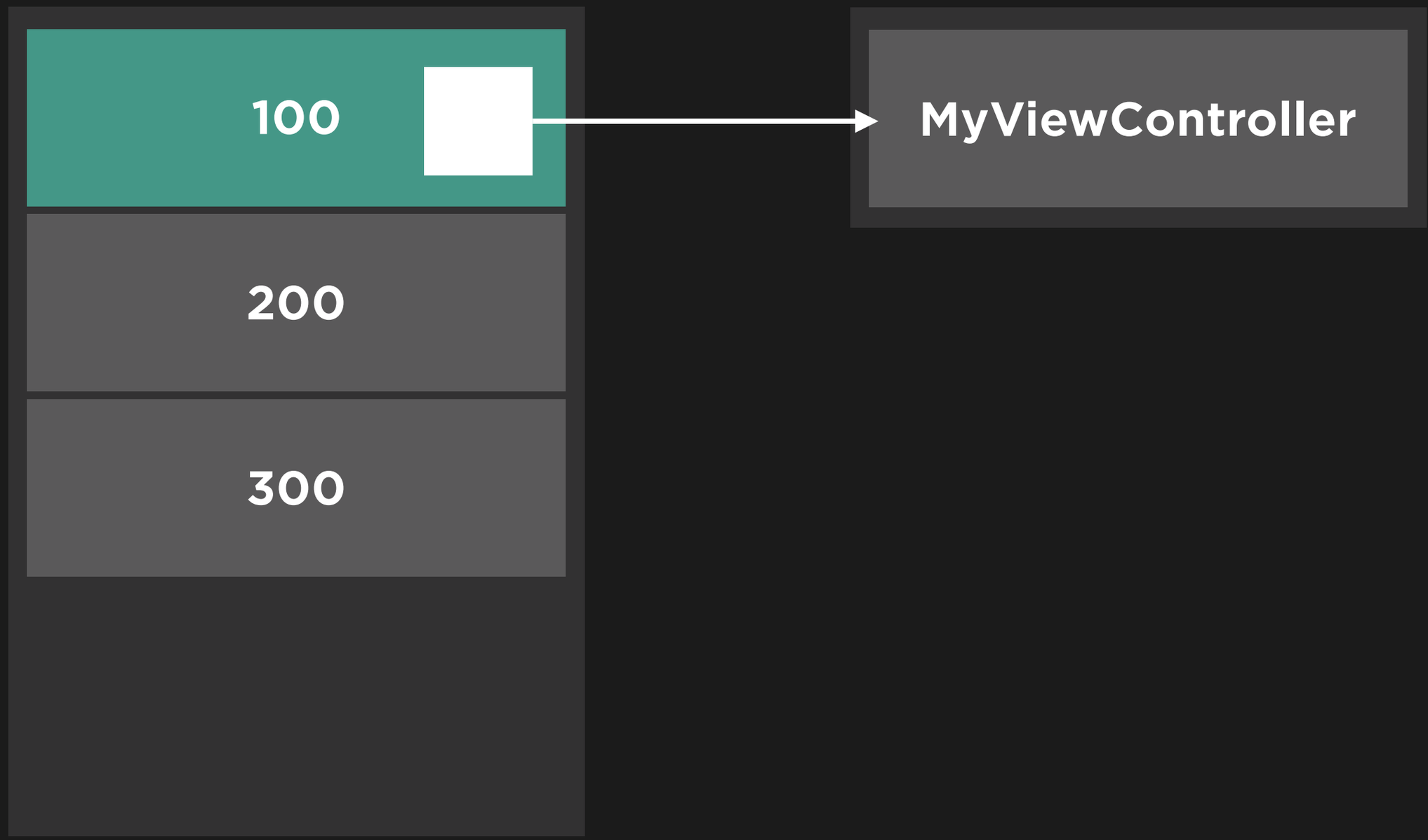
Component

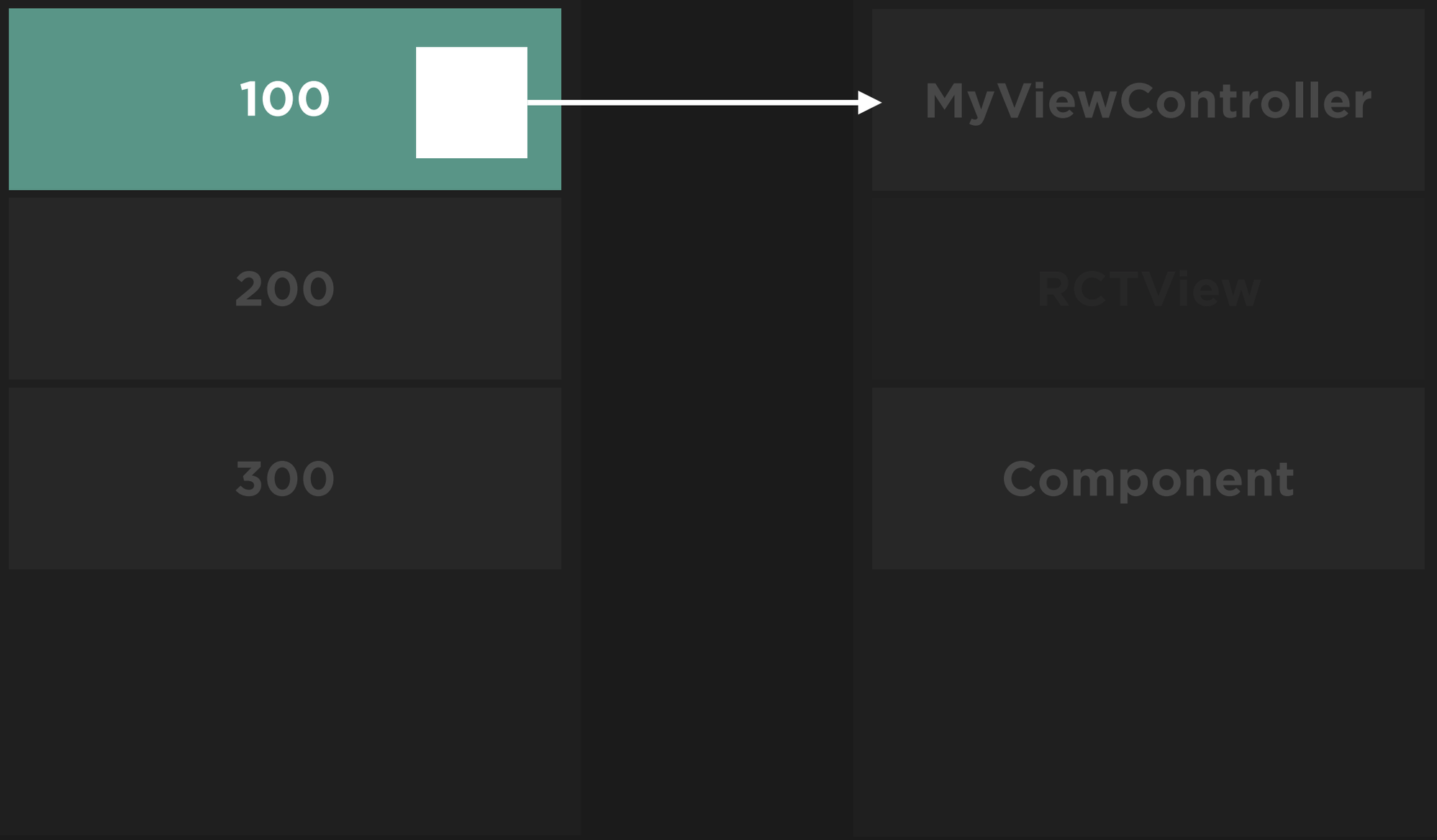
Component

Instance1

Instance 2


NativeModuleManager
MyViewController
MyViewController
RCTView
RCTView
Native
React
Component
Component
Instance1
Instance 2


100
200
300
MyViewController


100
MyViewController
200
RCTView
300
Component

```
NativeModules.MyComponentManager.doSomething(
this.props.reference,
‘hello world’);
```

MyComponentManager.swift

```
@objc public final func doSomething(
    reference: String,
    data: String) {

    let myView = referenceMap[reference]
    myView.doSomething(data)
}
```

MyComponentManager.swift

```
@objc public final func doSomething(
  reference: String,
  data: String) {

  let myView = referenceMap[reference]
  myView.doSomething(data)
}
```

MyComponentManager.swift

```
@objc public final func doSomething(
  reference: String,
  data: String) {
  let myView = referenceMap[reference]
  myView.doSomething(data)
}
```

MyComponent.swift

```
public final func doSomething(
   data: String) {
   print(“hello \(data)”)
}
```

So much boilerplate!

ReactViewController.swift

```
class ReactViewController {

  private func propsWithReference() ->
    [String: AnyObject]
  {
    var viewProps = props()
    viewProps[“reference"] =
      ObjectReference.reference(forObject: self)
    return viewProps
  }
```

ReactViewController.swift

```
class ReactViewController {

  private func propsWithReference() ->
    [String: AnyObject]
  {
    var viewProps = props()
    viewProps[“reference"] =
      ObjectReference.reference(forObject: self)
    return viewProps
  }
```

ReactViewController.swift

```
class ReactViewController {

  private func propsWithReference() ->
    [String: AnyObject]
  {
    var viewProps = props()
    viewProps[“reference"] =
      ObjectReference.reference(forObject: self)
    return viewProps
  }
```

ReactViewController.swift

```
class ReactViewController {

  private func propsWithReference() ->
    [String: AnyObject]
  {
    var viewProps = props()
    viewProps[“reference"] =
      ObjectReference.reference(forObject: self)
    return viewProps
  }
```

ReactViewController.swift

```
class ReactViewController {

  override open func viewDidLoad() {
      super.viewDidLoad()

      reactView.initializeReactView(
        componentName: componentName,
        props: propsWithReference())
}
```

```
SQS_CALLBACK_METHOD(
  doSomething:(nonnull NSString *)reference
    data:(nonnull NSString *)data,
  MyNativeWrapper,
  @selector(doSomething:), (@[data])
);
```

```
SQS_CALLBACK_METHOD(
  doSomething:(nonnull NSString *)reference
    data:(nonnull NSString *)data,
  MyNativeWrapper,
  @selector(doSomething:), (@[data])
);
```

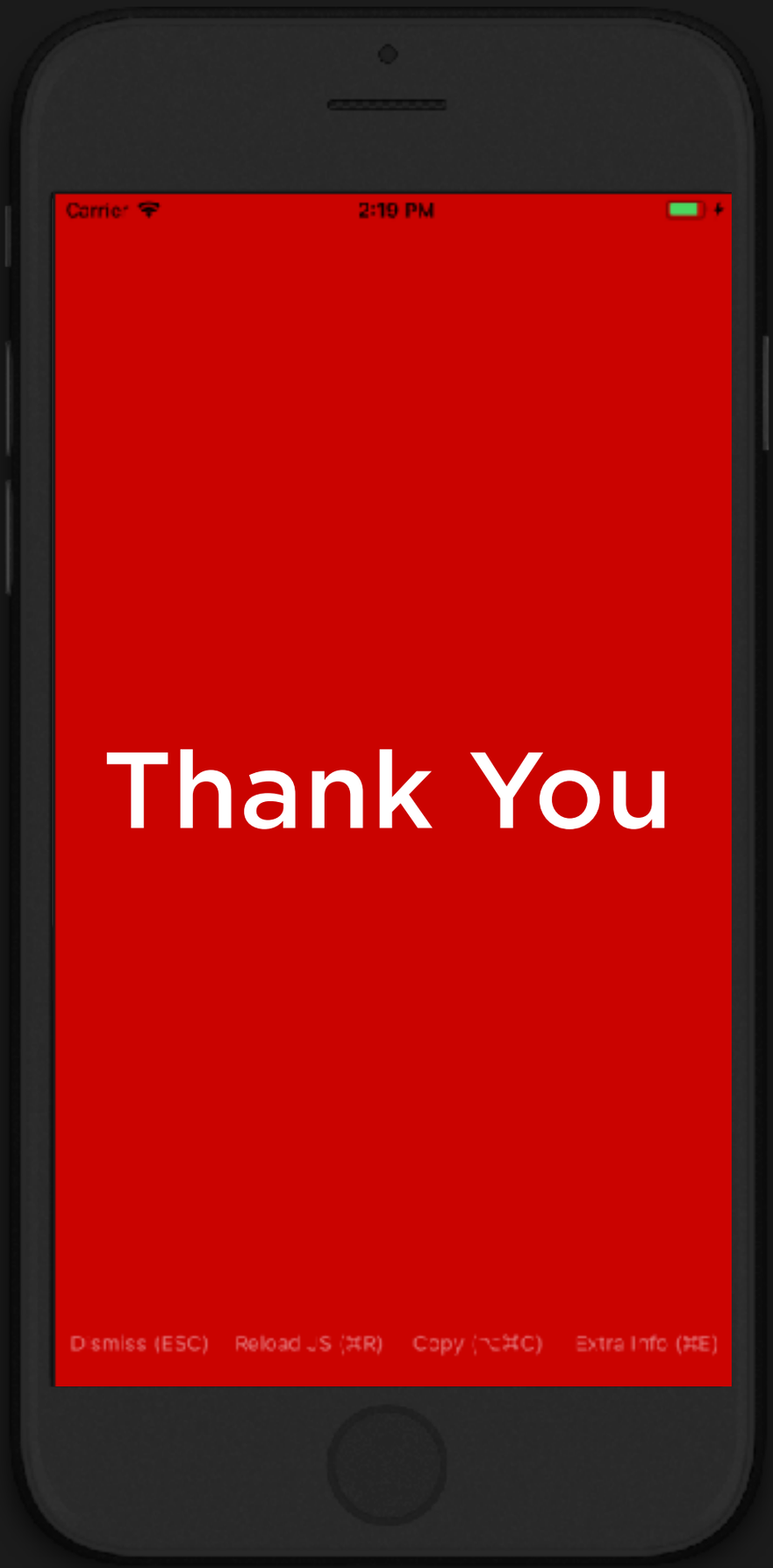
```
SQS_CALLBACK_METHOD(
  doSomething:(nonnull NSString *)reference
    data:(nonnull NSString *)data,
  MyNativeWrapper,
  @selector(doSomething:), (@[data])
);
```

```
SQS_CALLBACK_METHOD(
  doSomething:(nonnull NSString *)reference
    data:(nonnull NSString *)data,
  MyNativeWrapper,
  @selector(doSomething:), (@[data])
);
```

```
@implementation MyComponentManager
RCT_EXPORT_MODULE();

  SQS_CALLBACK_METHOD(
  doSomething:(nonnull NSString *)reference
    data:(nonnull NSString *)data,
  MyNativeWrapper,
  @selector(doSomething:), (@[data])
);
```

We've covered a lot

Architecture
Build a Component
Native Integration


Carrier
2:19 PM
Thank You
Dismiss (ESC)
Reload JS (⌘R)
Copy (⌥⌘C)
Extra Info (⌘E)

bscarano.github.io/crossplatform

@scarano

rscarano@squarespace.com