



RN

Writing custom Native Modules

BALA KRISHNA RAGALA



Native Modules

Custom code from native platform

What are Native Modules

- Targeted platform API based code (modules)
- Some Examples
 - ImageProcessing
 - Camera access
 - Database
 - Playing videos and audio
 - OAuth provider for authenticating users

Why Native Modules

- Reusability
- Extensibility
- Multithreaded capabilities
- High performance code

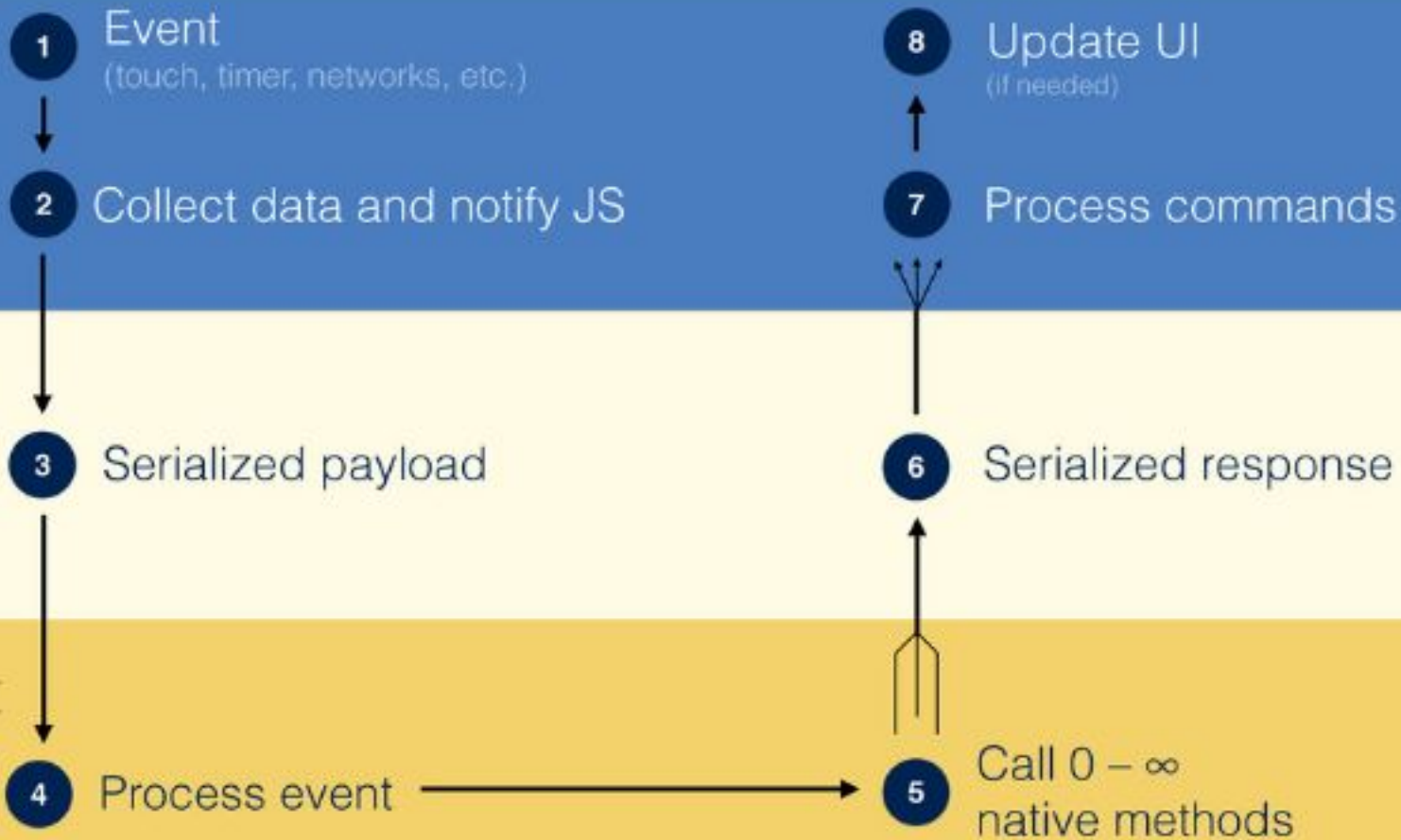
What Facebook says

We designed React Native such that it is possible for you to write real native code and have access to the full power of the platform. **This is a more advanced feature and we don't expect it to be part of the usual development process, however it is essential that it exists.** If React Native doesn't support a native feature that you need, you should be able to build it yourself.

Writing Custom Native Modules

Be a creator :-)

Native



What we will build & Learn

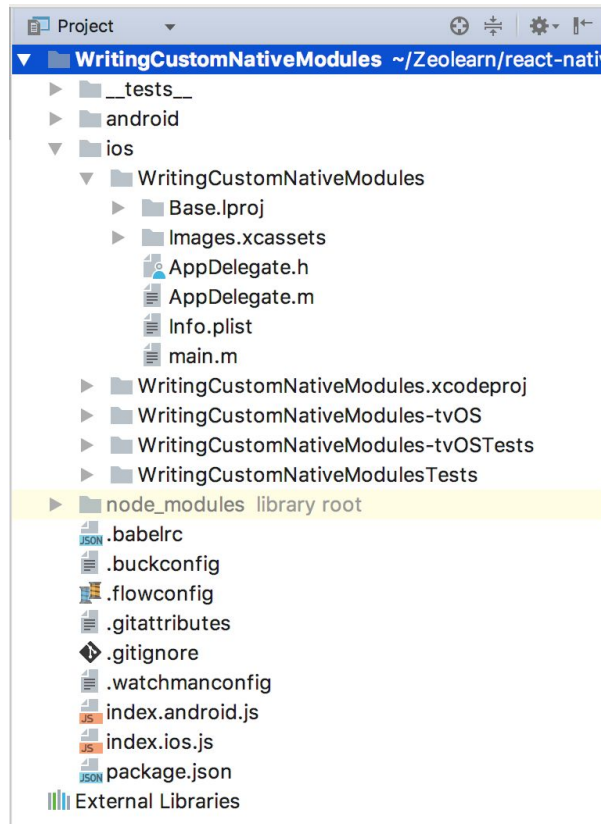
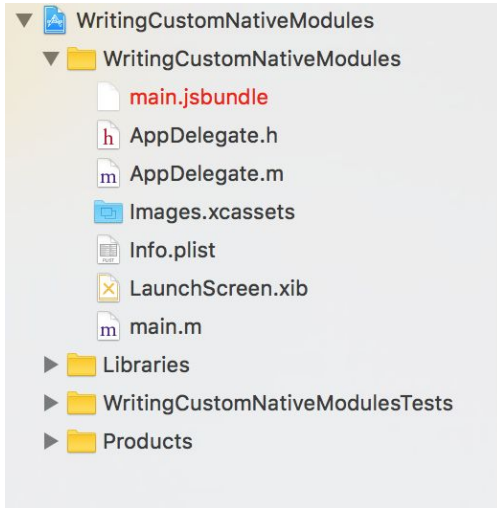
- Build
 - Library Manager that allow the user to select an image from device's media library
- Learn
 - Exposing both native methods and constants
 - Different methods of communicating between JavaScript and native code, including callbacks, promises, and events

Let's code - iOS Time

Time to use iOS & JS coding skills

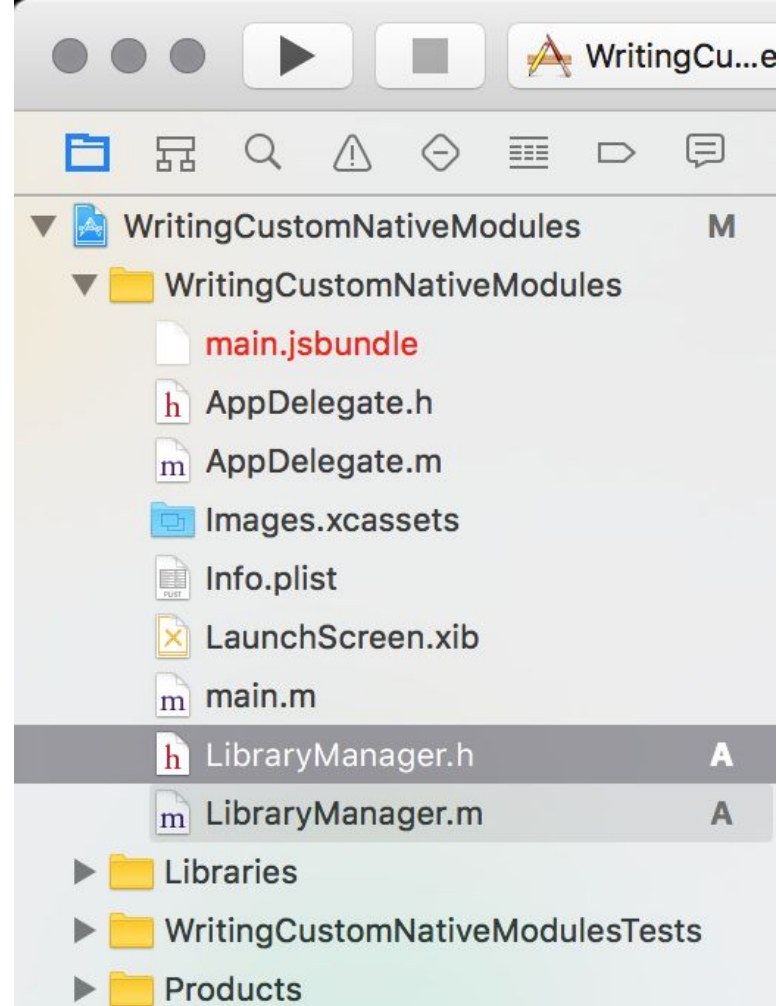
Setting it up

- react-native init WritingCustomNativeModules
- Expand ios folder and look for xcode project WritingCustomNativeModules.xcodeproj
- Double click the xcodeproj to open it in xcode



Setting it up - Xcode

- Double click the xcodeproj to open it in xcode
- Add header and implementation files named as LibraryManager



Setting it up - Xcode - RCT

```
// LibraryManager.h  
#import "React/RCTBridgeModule.h"
```

```
@interface LibraryManager : NSObject <RCTBridgeModule>  
@end
```

```
// LibraryManager.m
```

```
#import "LibraryManager.h"
```

```
@implementation LibraryManager
```

```
RCT_EXPORT_MODULE();
```

```
@end
```

RCT from source*

```
/**
 * Provides the interface needed to register a bridge module.
 */
@protocol RCTBridgeModule <NSObject>

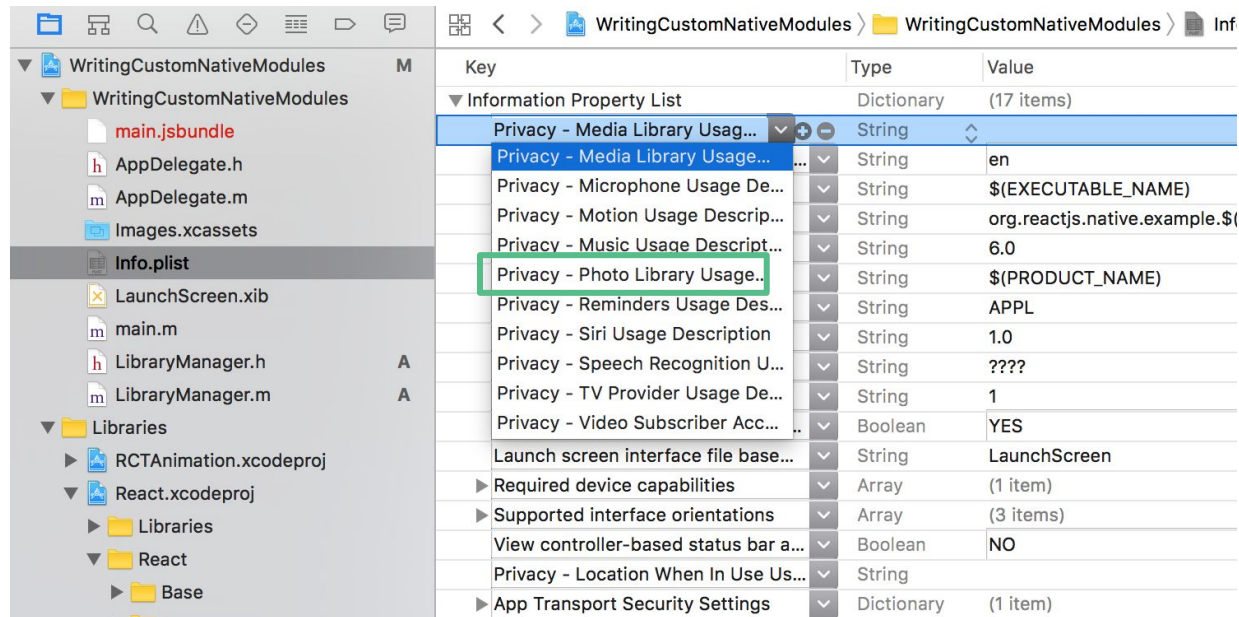
/**
 * Place this macro in your class implementation to automatically register
 * your module with the bridge when it loads. The optional js_name argument
 * will be used as the JS module name. If omitted, the JS module name will
 * match the Objective-C class name.
 */
#define RCT_EXPORT_MODULE(js_name) \
RCT_EXTERN void RCTRegisterModule(Class); \
+ (NSString *)moduleName { return @"#js_name; } \
+ (void)load { RCTRegisterModule(self); }
```

Runtime Permission - iOS 10

As of iOS 10, if an application will access the user's image library, it needs to provide an explanation for this in the **Info.plist** file.

To do this, select the **Info.plist** file on the left-hand side in Xcode.

Then, add this new value by selecting **Privacy - Photo Library Usage Description** and providing a brief explanation



Setting it up - JS realm

1. Import NativeModules from react-native library
2. Destructure required native module

```
/**  
 * Sample React Native App  
 * https://github.com/facebook/react-native  
 * @flow  
 */
```

```
import React, {Component} from 'react';  
import {  
  AppRegistry,  
  StyleSheet,  
  Text,  
  View,  
  NativeModules  
} from 'react-native';
```

```
const {LibraryManager} = NativeModules;
```

```
console.log(NativeModules);
```

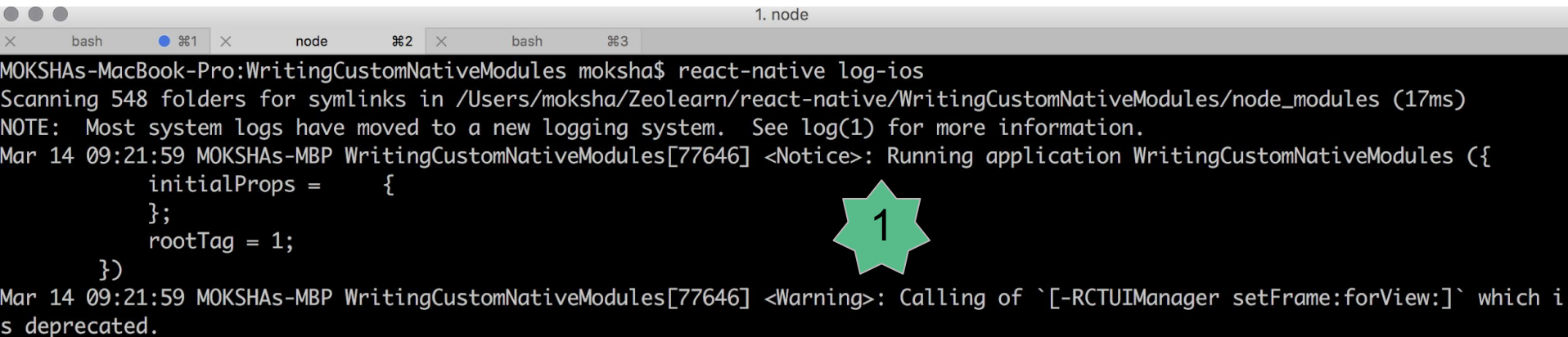
Debug Tip

You can view JS console.* statements on the terminal where ios code run is initiated with command

react-native log-ios



```
RCTTabBar:
{ Manager: 'TabBarManager',
  NativeProps:
    { unselectedTintColor: 'UIColor',
      tint_color: 'UIColor',
      unselectedItemTintColor: 'UIColor',
      translucent: 'BOOL',
      barTintColor: 'UIColor',
      itemPositioning: 'UITabBarItemPositioning' },
  Constants: [Getter/Setter],
  Commands: [Getter/Setter] } },
LibraryManager: null }
Mar 14 09:25:15 MOKSHAS-MBP WritingCustomNativeModules[77646] <No
initialProps = {
```



```
1. node
MOKSHAS-MacBook-Pro:WritingCustomNativeModules moksha$ react-native log-ios
Scanning 548 folders for symlinks in /Users/moksha/Zeolearn/react-native/WritingCustomNativeModules/node_modules (17ms)
NOTE: Most system logs have moved to a new logging system. See log(1) for more information.
Mar 14 09:21:59 MOKSHAS-MBP WritingCustomNativeModules[77646] <Notice>: Running application WritingCustomNativeModules ({
  initialProps = {
  };
  rootTag = 1;
})
Mar 14 09:21:59 MOKSHAS-MBP WritingCustomNativeModules[77646] <Warning>: Calling of `[-RCTUIManager setFrame:forView:]` which is deprecated.
```


Let's code - Adding functionality

Time to use iOS coding skills

Exporting Methods

We can export a method of a module by using the `RCT_EXPORT_METHOD` macro

```
RCT_EXPORT_METHOD(selectImage)
{
  RCTLogInfo(@"Selecting image...");
}
```

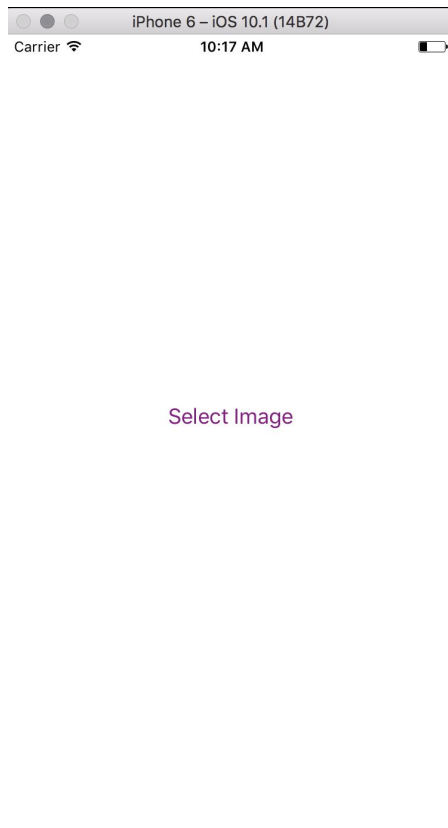
```
/**
 * Wrap the parameter line of your method implementation with this macro to
 * expose it to JS. By default the exposed method will match the first part of
 * the Objective-C method selector name (up to the first colon). Use
 * RCT_REMAP_METHOD to specify the JS name of the method.
 *
 * For example, in ModuleName.m:
 *
 * - (void)doSomething:(NSString *)aString withA:(NSInteger)a andB:(NSInteger)b
 * { ... }
 *
 * becomes
 *
 * RCT_EXPORT_METHOD(doSomething:(NSString *)aString
 *                   withA:(NSInteger)a
 *                   andB:(NSInteger)b)
 * { ... }
 *
 * and is exposed to JavaScript as `NativeModules.ModuleName.doSomething`.
 *
 * ## Promises
 *
 * Bridge modules can also define methods that are exported to JavaScript as
 * methods that return a Promise, and are compatible with JS async functions.
 *
 * Declare the last two parameters of your native method to be a resolver block
 * and a rejecter block. The resolver block must precede the rejecter block.
 *
 * For example:
 *
 * RCT_EXPORT_METHOD(doSomethingAsync:(NSString *)aString
 *                   resolver:(RCTPromiseResolveBlock)resolve
 *                   rejecter:(RCTPromiseRejectBlock)reject)
 * { ... }
 *
 * Calling `NativeModules.ModuleName.doSomethingAsync(aString)` from
 * JavaScript will return a promise that is resolved or rejected when your
 * native method implementation calls the respective block.
 */
#define RCT_EXPORT_METHOD(method) \
  RCT_REMAP_METHOD(, method)
```

Let's Run

Whenever there is a change in native code normal JS refresh is not sufficient

You need to rebuild the code, else you end up with error

Run react-native run-ios again



Let's see log

Notice LibraryManager native module object, it has selectImage function

Now clicking button puts a log
"selecting image..."

```
Commands: [Getter/Setter] } },
LibraryManager: { selectImage: { [Function: fn] type: 'async' } } }
Mar 14 10:20:56 MOKSHAS-MBP WritingCustomNativeModules[80405] <Notice>: Running a
initialProps = {
};
```

```
ppParams: {"rootTag":1,"initialProps":{}}. __DEV__ === true, development-level warning are ON
F
Mar 14 10:22:14 MOKSHAS-MBP WritingCustomNativeModules[80405] <Notice>: Selecting image...
```

Code native functionality - 1

```
RCT_EXPORT_METHOD(selectImage)
{
    RCTLogInfo(@"Selecting image...");

    UIImagePickerController *picker = [[UIImagePickerController alloc] init];

    picker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    picker.mediaTypes = @[(NSString *)kUTTypeImage];
    picker.modalPresentationStyle = UIModalPresentationCurrentContext;

    picker.delegate = self;

    UIViewController *root = [[[[UIApplication sharedApplication] delegate] window] rootViewController];

    [root presentViewController:picker animated:YES completion:nil];
}
```

Code native functionality - 2

```
#import <React/RCTBridgeModule.h>
#import <UIKit/UIKit.h>

@interface LibraryManager : NSObject <RCTBridgeModule, UINavigationControllerDelegate, UIImagePickerControllerDelegate>
@end

- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    NSString *fileName = [[[NSUUID UUID] UUIDString] stringByAppendingString:@"jpg"];
    NSString *path = [[NSTemporaryDirectory()]stringByStandardizingPath] stringByAppendingPathComponent:fileName];
    UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
    NSData *data = UIImageJPEGRepresentation(image, 0);
    [data writeToFile:path atomically:YES];
    NSURL *fileURL = [NSURL fileURLWithPath:path];
    NSString *filePath = [fileURL absoluteString];

    RCTLog(@"%@", filePath);

    [picker dismissViewControllerAnimated:YES completion:nil];
}
```

It's time for messaging

React-native to JS, JS to React-native

Communication between JS & RN

- Callback
- Promise
- Events

Communication with Callback

JS : Hey react-native, call me back once you are done !!!

Code native functionality - 3

```
RCT_EXPORT_METHOD(selectImage:(RCTResponseSenderBlock)callback)
{
    RCTLogInfo(@"Selecting image...");
    self.callback = callback;

    UIImagePickerController *picker = [[UIImagePickerController alloc] init];

    picker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    picker.mediaTypes = @[(NSString *)kUTTypeImage];
    picker.modalPresentationStyle = UIModalPresentationCurrentContext;

    picker.delegate = self;

    UIViewController *root = [[[[UIApplication sharedApplication] delegate] window] rootViewController];

    [root presentViewController:picker animated:YES completion:nil];
}

- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    NSString *fileName = [[[NSUUID UUID] UUIDString] stringByAppendingString:@"jpg"];
    NSString *path = [[NSTemporaryDirectory()]stringByStandardizingPath] stringByAppendingString:pathComponent:fileName];
    UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
    NSData *data = UIImageJPEGRepresentation(image, 0);
    [data writeToFile:path atomically:YES];
    NSURL *fileURL = [NSURL fileURLWithPath:path];
    NSString *filePath = [fileURL absoluteString];

    RCTLog(@"%@", filePath);

    self.callback([filePath]);

    [picker dismissViewControllerAnimated:YES completion:nil];
}
```

Code native functionality - 4

```
export default class WritingCustomNativeModules extends Component {  
  constructor(props) {  
    super(props);  
    this.state = {};  
    this.onSelectImage = this.onSelectImage.bind(this);  
  }
```

```
  onSelectImage() {  
    LibraryManager.selectImage((filePath) => {  
      console.log("onSelectImage", filePath);  
      this.setState({selectedImageUrl: filePath})  
    });  
  }  
}
```

■ ■ ■

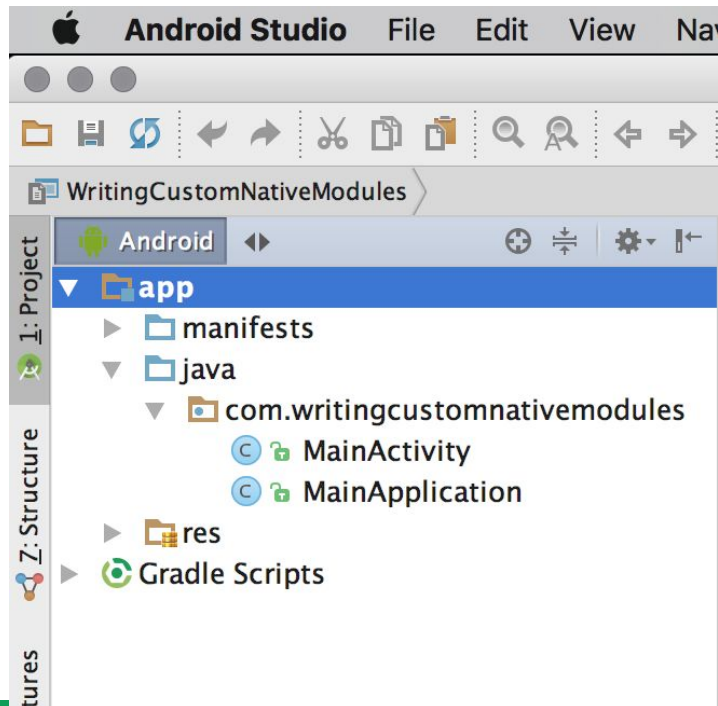
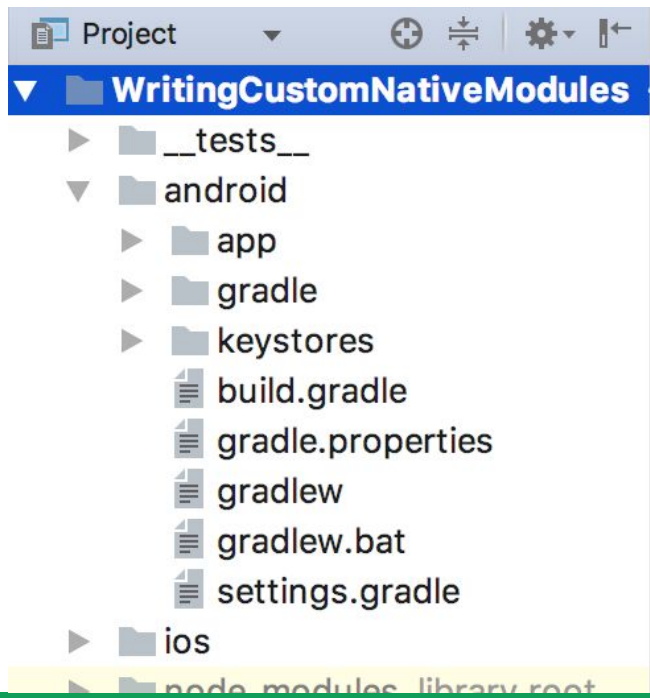
```
render() {  
  return (  
    <View style={[[globalStyles.COMMON_STYLES.pageContainer, styles.container]]}>  
      {this.renderProfileImage()}  
    </View>  
  );  
}
```

Let's code - Android Time

Time to use android & JS coding skills

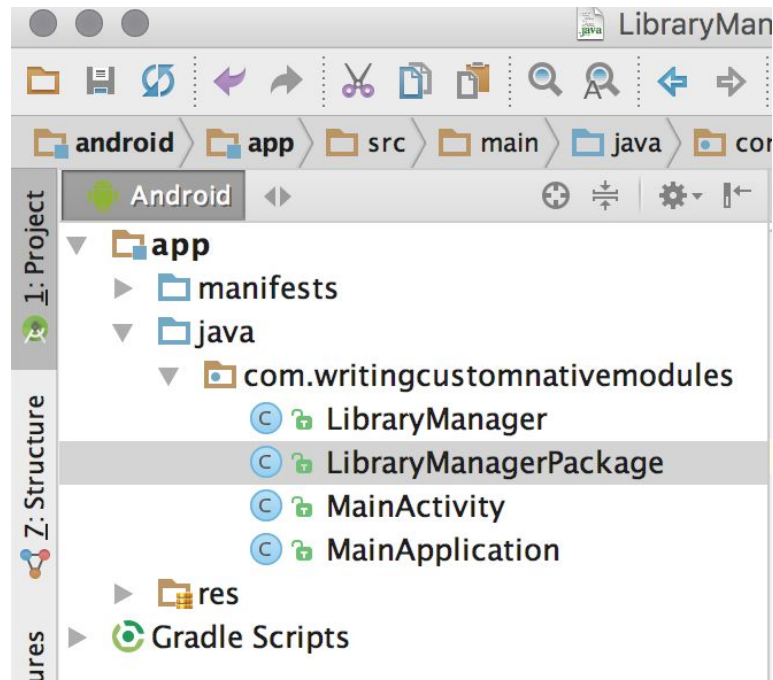
Setting it up

- Open up android studio and open android project



Setting it up - AS

- Add LibraryManager.java and LibraryManagerPackaga.java



Setting it up - Android - React

```
public class LibraryManager extends ReactContextBaseJavaModule {  
    1  
}
```

```
public class LibraryManagerPackage implements ReactPackage {  
    @Override  
    public List<NativeModule> createNativeModules(ReactApplicationContext reactContext) {  
        List<NativeModule> nativeModules = new ArrayList<>();  
        nativeModules.add(new LibraryManager(reactContext));  
        return nativeModules;  
    }  
}
```

```
public class MainApplication extends Application implements ReactApplication {  
  
    private final ReactNativeHost mReactNativeHost = new ReactNativeHost(this) {  
        @Override  
        protected List<ReactPackage> getPackages() {  
            return Arrays.<ReactPackage>asList(  
                new MainReactPackage(),  
                new LibraryManagerPackage()  
            );  
        }  
    };  
}
```

React* from source

```
/**  
 * Base class for Catalyst native modules that require access to the {@link ReactContext}  
 * instance.  
 */
```

```
public abstract class ReactContextBaseJavaModule extends BaseJavaModule {
```

```
/**  
 * Base class for Catalyst native modules whose implementations are written in Java. Default  
 * implementations for {@link #initialize} and {@link #onCatalystInstanceDestroy} are provided for  
 * convenience. Subclasses which override these don't need to call {@code super} in case of  
 * overriding those methods as implementation of those methods is empty.  
 *  
 * BaseJavaModules can be linked to Fragments' lifecycle events, {@link CatalystInstance} creation  
 * and destruction, by being called on the appropriate method when a life cycle event occurs.  
 *  
 * Native methods can be exposed to JS with {@link ReactMethod} annotation. Those methods may  
 * only use limited number of types for their arguments:  
 * 1/ primitives (boolean, int, float, double  
 * 2/ {@link String} mapped from JS string  
 * 3/ {@link ReadableArray} mapped from JS Array  
 * 4/ {@link ReadableMap} mapped from JS Object  
 * 5/ {@link Callback} mapped from js function and can be used only as a last parameter or in the  
 * case when it express success & error callback pair as two last arguments respectively.  
 *  
 * All methods exposed as native to JS with {@link ReactMethod} annotation must return  
 * {@code void}.  
 *  
 * Please note that it is not allowed to have multiple methods annotated with {@link ReactMethod}  
 * with the same name.  
 */  
public abstract class BaseJavaModule implements NativeModule {
```


React* from source

```
/**
 * Main interface for providing additional capabilities to the catalyst framework by couple of
 * different means:
 * 1) Registering new native modules
 * 2) Registering new JS modules that may be accessed from native modules or from other parts of the
 * native code (requiring JS modules from the package doesn't mean it will automatically be included
 * as a part of the JS bundle, so there should be a corresponding piece of code on JS side that will
 * require implementation of that JS module so that it gets bundled)
 * 3) Registering custom native views (view managers) and custom event types
 * 4) Registering natively packaged assets/resources (e.g. images) exposed to JS
 *
 * TODO(6788500, 6788507): Implement support for adding custom views, events and resources
 */
public interface ReactPackage {
```

Setting it up - JS realm

1. Import NativeModules from react-native library
2. Destructure required native module

```
/**  
 * Sample React Native App  
 * https://github.com/facebook/react-native  
 * @flow  
 */
```

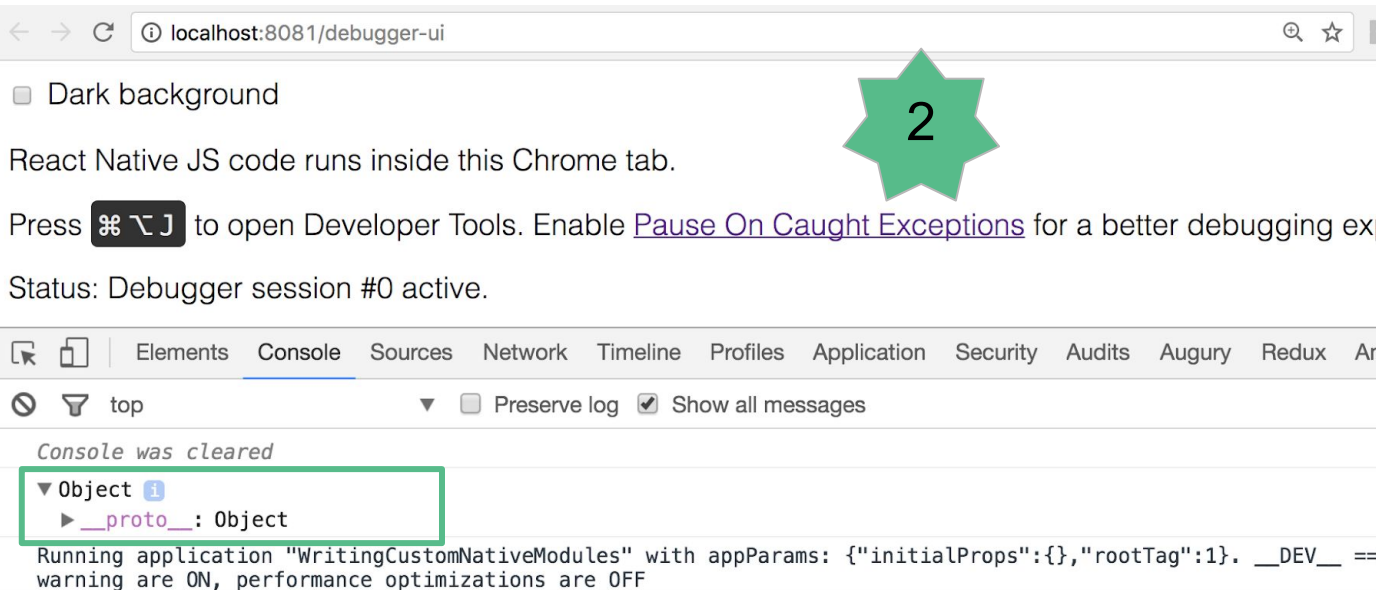
```
import React, {Component} from 'react';  
import {  
  AppRegistry,  
  StyleSheet,  
  Text,  
  View,  
  NativeModules  
} from 'react-native';
```

```
const {LibraryManager} = NativeModules;
```

```
console.log(NativeModules);
```

Debug Tip

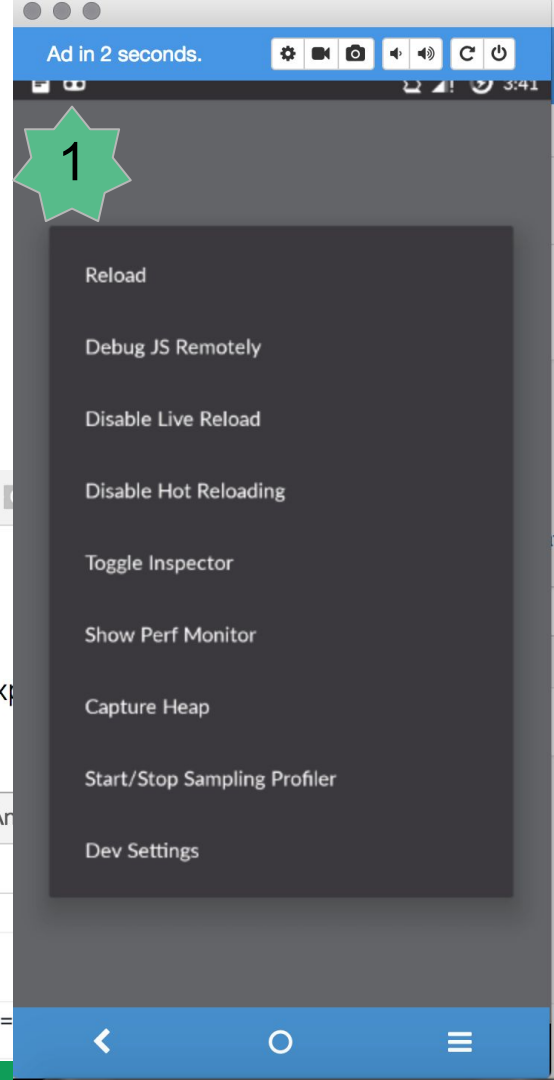
You can view JS console.* statements on chrome console by enabling JS remote debugging



The screenshot shows the Chrome DevTools interface. The address bar displays `localhost:8081/debugger-ui`. The left sidebar has a `Dark background` toggle. The main content area states: "React Native JS code runs inside this Chrome tab. Press `⌘ ⇧ J` to open Developer Tools. Enable [Pause On Caught Exceptions](#) for a better debugging experience. Status: Debugger session #0 active." The top of the DevTools panel shows tabs for Elements, Console, Sources, Network, Timeline, Profiles, Application, Security, Audits, Augury, Redux, and Architecture. The Console tab is active, showing a message "Console was cleared" and a log entry for an "Object" with a `__proto__` property. A green box highlights the log entry. The bottom of the DevTools panel shows the status bar with the text: "Running application 'WritingCustomNativeModules' with appParams: {'initialProps': {}, 'rootTag': 1}. __DEV__ == true, warning are ON, performance optimizations are OFF".

1

2



Let's code - Adding functionality

Time to use android coding skills

Exporting Methods

We can export a method of a module by using the `@ReactMethod` annotation

```
/**
 * Annotation which is used to mark methods that are exposed to React Native.
 *
 * This applies to derived classes of {@link BaseJavaModule}, which will generate a list
 * of exported methods by searching for those which are annotated with this annotation
 * and adding a JS callback for each.
 */
@Retention(RUNTIME)
public @interface ReactMethod {

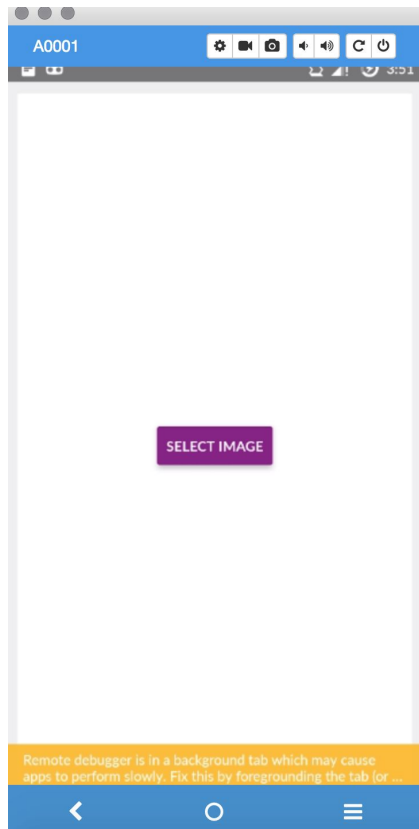
    @ReactMethod
    public void selectImage() {
        Activity currentActivity = getCurrentActivity();
        Intent libraryIntent = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        currentActivity.startActivityForResult(libraryIntent, 1);
    }
}
```

Let's Run

Whenever there is a change in native code normal JS refresh is not sufficient

You need to rebuild the code, else you end up with error

Run react-native run-android again



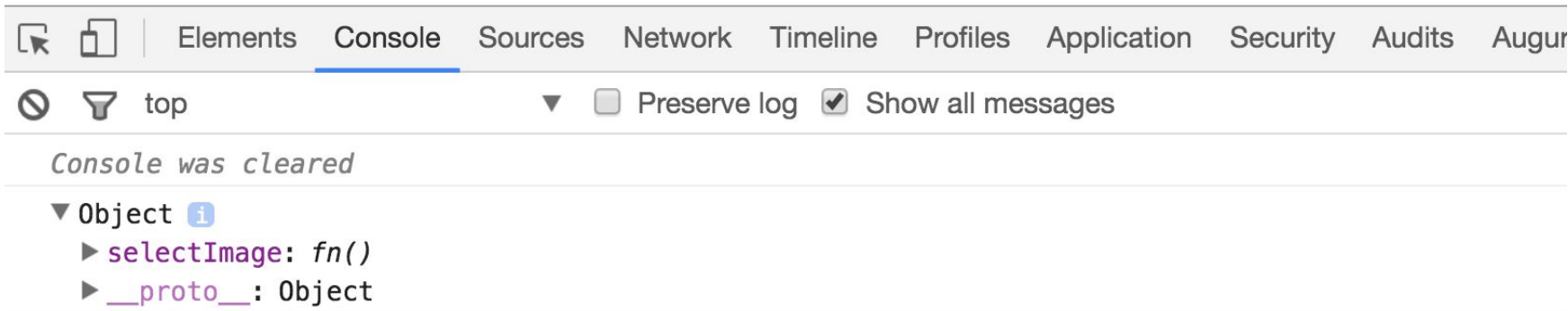
Let's see log

Notice LibraryManager native module object, it has selectImage function

Now clicking button open image picker

Press `⌘ ⇧ J` to open Developer Tools. Enable [Pause On Caught Exceptions](#) for a better de

Status: Debugger session #0 active.



Code native functionality - 1

@ReactMethod

```
public void selectImage() {  
    Activity currentActivity = getCurrentActivity();  
    Intent libraryIntent = new Intent(Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);  
    currentActivity.startActivityForResult(libraryIntent, 1);  
}
```


Code native functionality - 2

```
public class LibraryManager extends ReactContextBaseJavaModule implements ActivityEventListener {

    public LibraryManager(ReactApplicationContext reactContext) {
        super(reactContext);
        reactContext.addActivityEventListener(this);
    }

    @Override
    public void onActivityResult(Activity activity, int requestCode, int resultCode, Intent data) {
        String filePath = data.getDataString();
    }

    @Override
    public void onNewIntent(Intent intent) {

    }
}
```

It's time for messaging

React-native to JS, JS to React-native

Communication between JS & RN

- Callback
- Promise
- Events

Communication with Callback

JS : Hey react-native, call me back once you are done !!!

Code native functionality - 3

```
RCT_EXPORT_METHOD(selectImage:(RCTResponseSenderBlock)callback)
{
    RCTLogInfo(@"Selecting image...");
    self.callback = callback;

    UIImagePickerController *picker = [[UIImagePickerController alloc] init];

    picker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
    picker.mediaTypes = @[(NSString *)kUTTypeImage];
    picker.modalPresentationStyle = UIModalPresentationCurrentContext;

    picker.delegate = self;

    UIViewController *root = [[[[UIApplication sharedApplication] delegate] window] rootViewController];

    [root presentViewController:picker animated:YES completion:nil];
}

- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    NSString *fileName = [[[NSUUID UUID] UUIDString] stringByAppendingString:@"jpg"];
    NSString *path = [[NSTemporaryDirectory()]stringByStandardizingPath] stringByAppendingString:pathComponent:fileName];
    UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
    NSData *data = UIImageJPEGRepresentation(image, 0);
    [data writeToFile:path atomically:YES];
    NSURL *fileURL = [NSURL fileURLWithPath:path];
    NSString *filePath = [fileURL absoluteString];

    RCTLog(@"%@", filePath);

    self.callback([filePath]);

    [picker dismissViewControllerAnimated:YES completion:nil];
}
```

Code native functionality - 4

```
export default class WritingCustomNativeModules extends Component {  
  constructor(props) {  
    super(props);  
    this.state = {};  
    this.onSelectImage = this.onSelectImage.bind(this);  
  }
```

```
  onSelectImage() {  
    LibraryManager.selectImage((filePath) => {  
      console.log("onSelectImage", filePath);  
      this.setState({selectedImageUrl: filePath})  
    });  
  }  
}
```

■ ■ ■

```
render() {  
  return (  
    <View style={[[globalStyles.COMMON_STYLES.pageContainer, styles.container]]}>  
      {this.renderProfileImage()}  
    </View>  
  );  
}
```