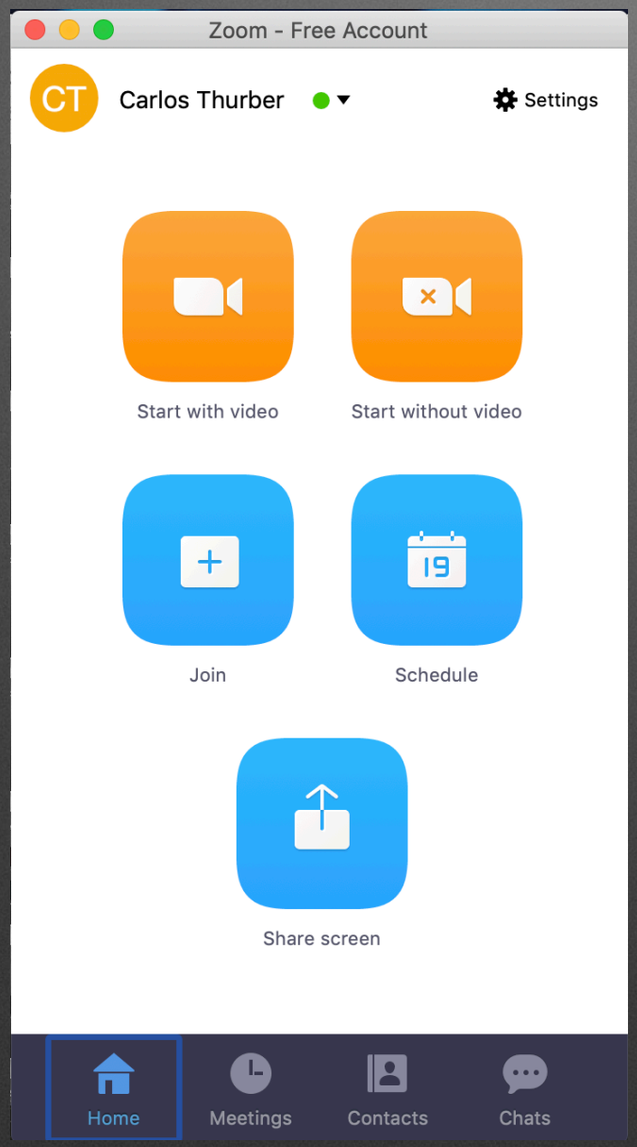
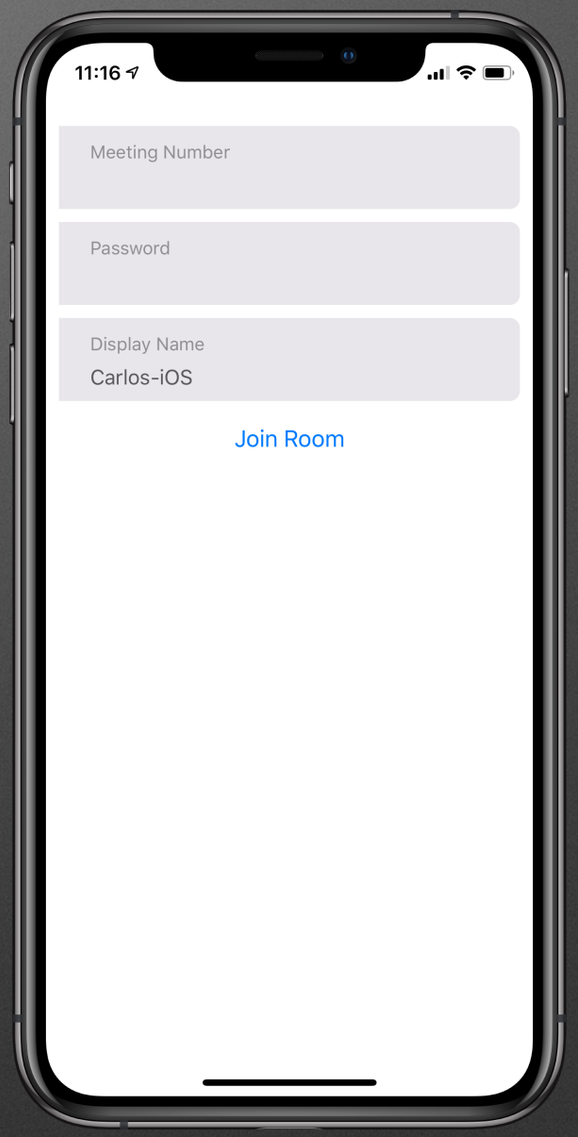
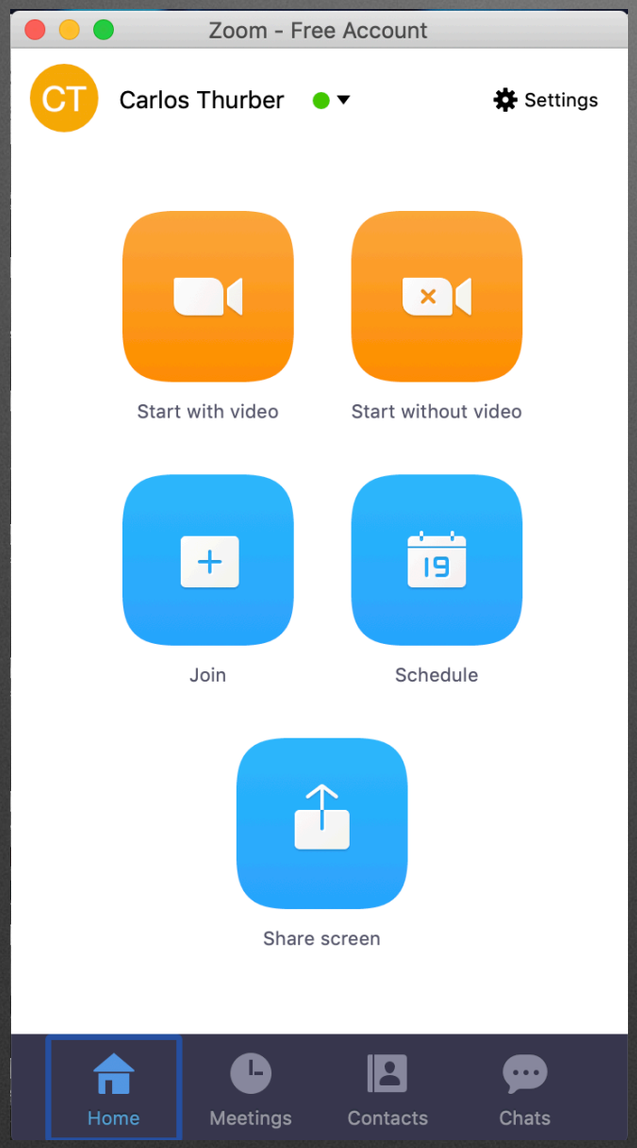
Implementing videocalls in React-Native with Zoom SDK

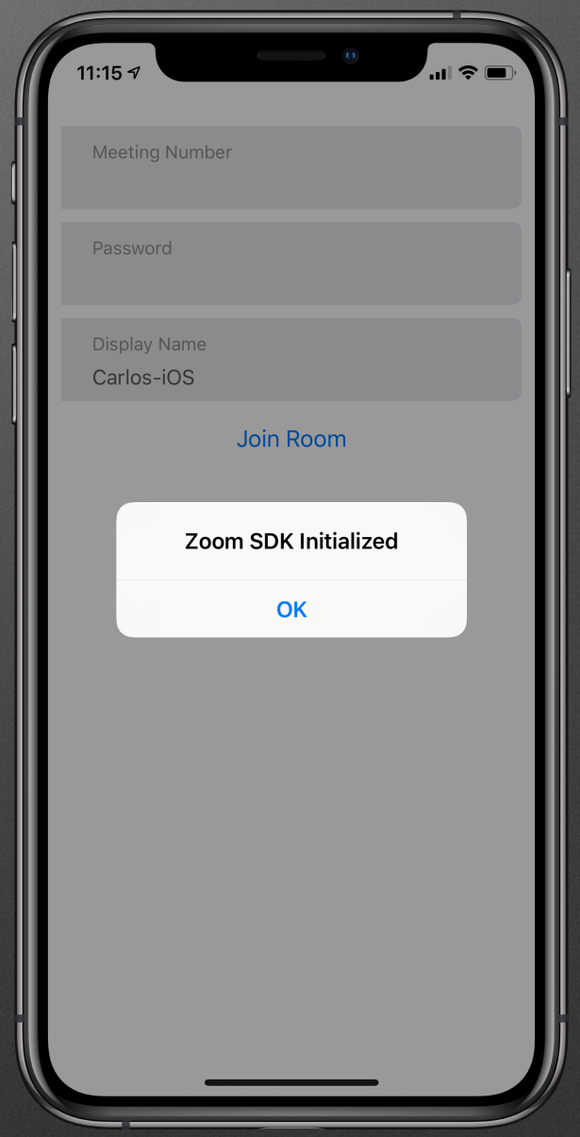
Demo


Zoom - Free Account
CT
Carlos Thurber
Settings
Start with video
Start without video
Join
Schedule
Share screen
Home
Meetings
Contacts
Chats

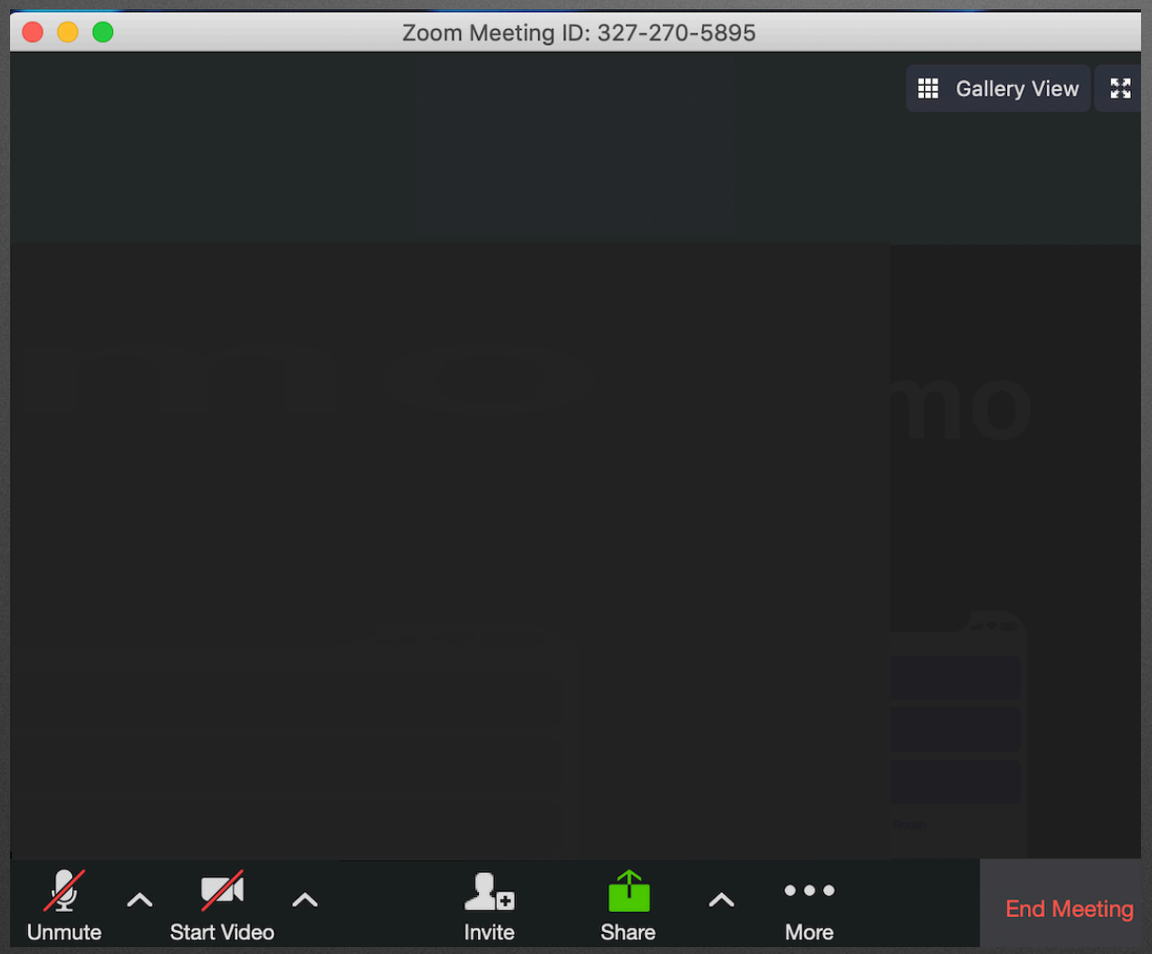

11:16
Meeting Number
Password
Display Name
Carlos-iOS
Join Room

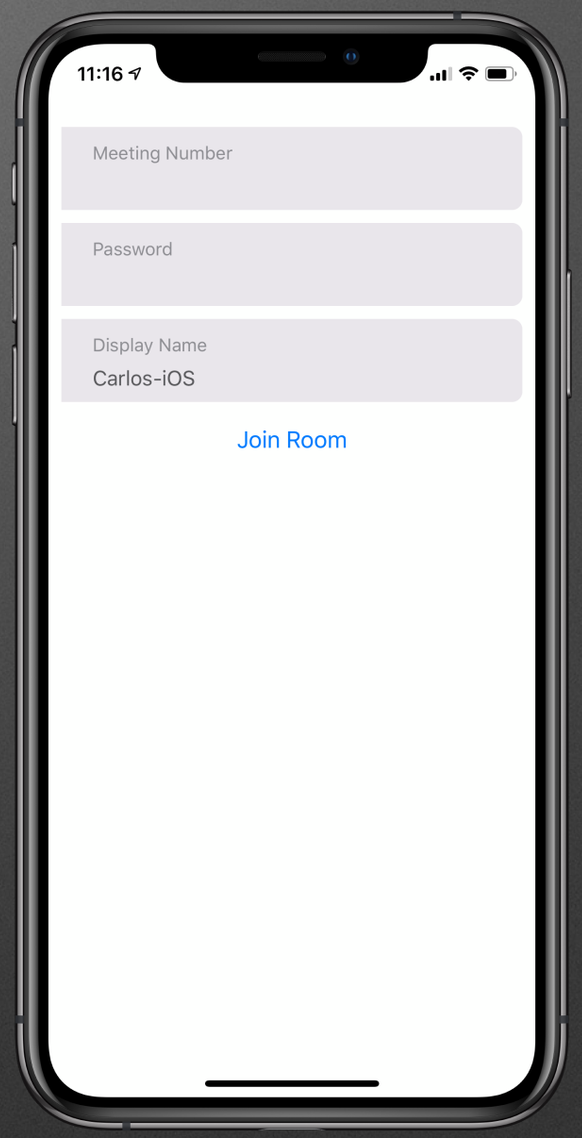
Demo


Zoom - Free Account
CT
Carlos Thurber
Settings
Start with video
Start without video
Join
Schedule
Share screen
Home
Meetings
Contacts
Chats

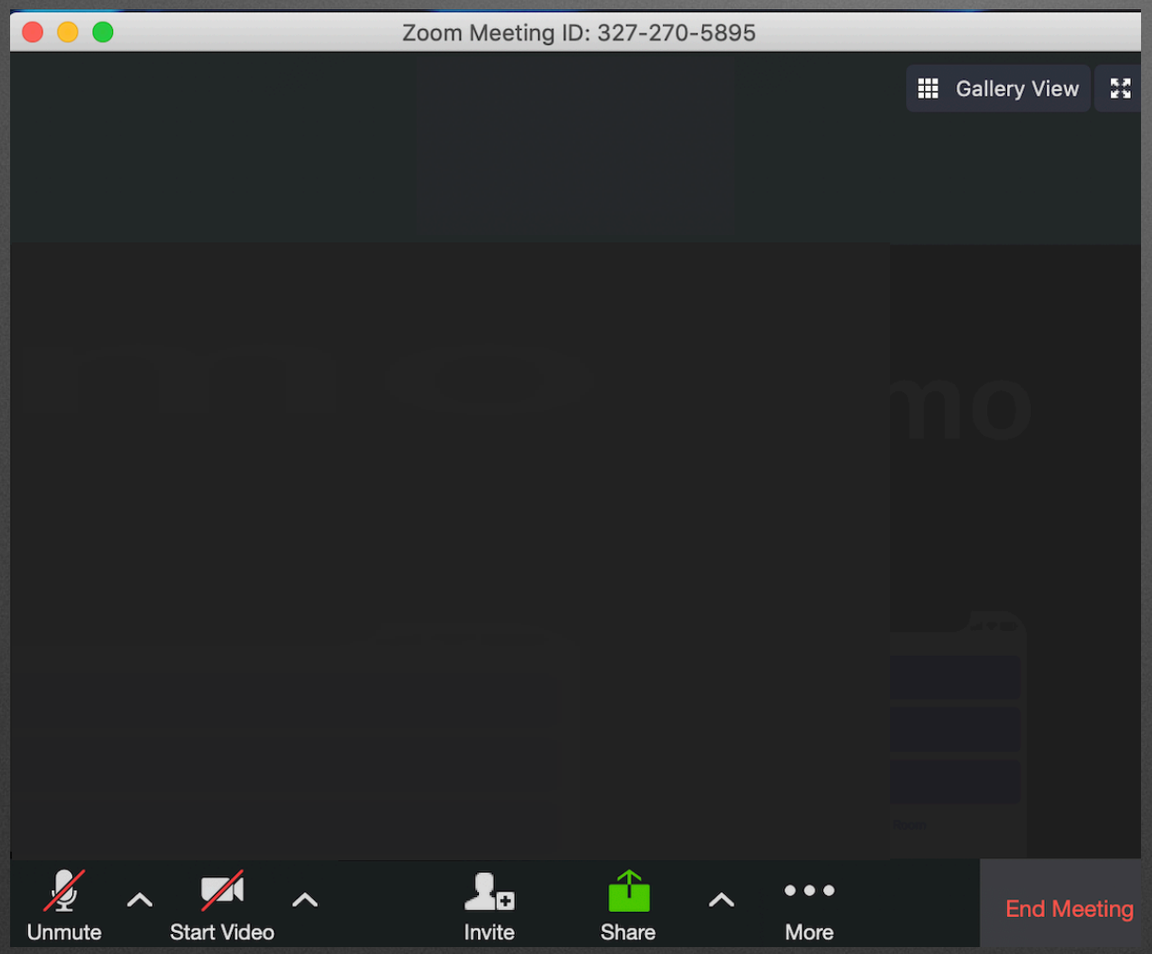

11:15
Meeting Number
Password
Display Name
Carlos-iOS
Join Room
Zoom SDK Initialized
OK

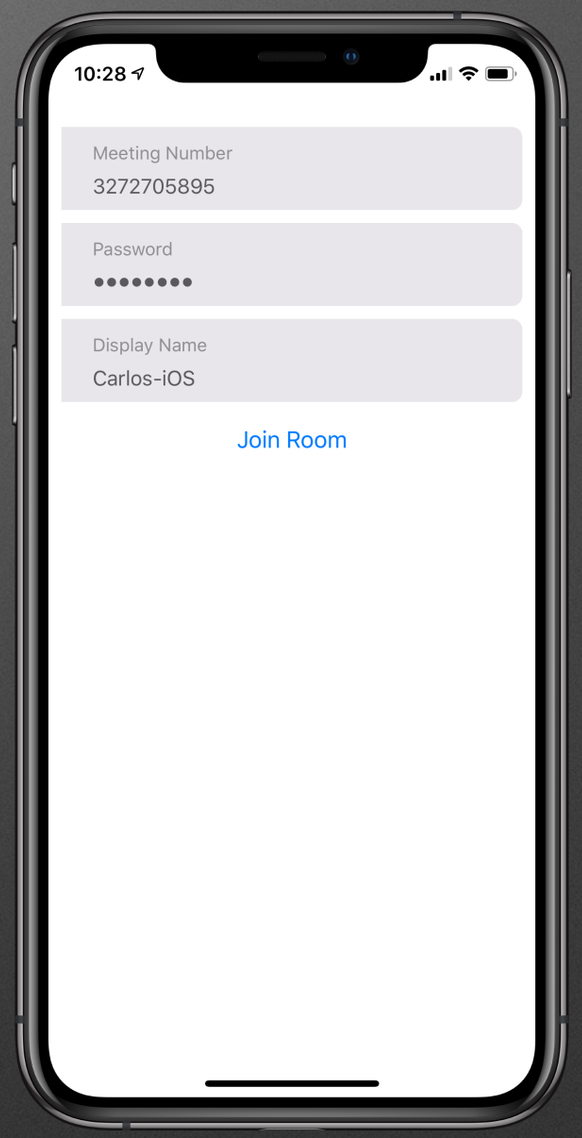
Demo


Zoom Meeting ID: 327-270-5895
Gallery View
Unmute
Start Video
Invite
Share
More
End Meeting

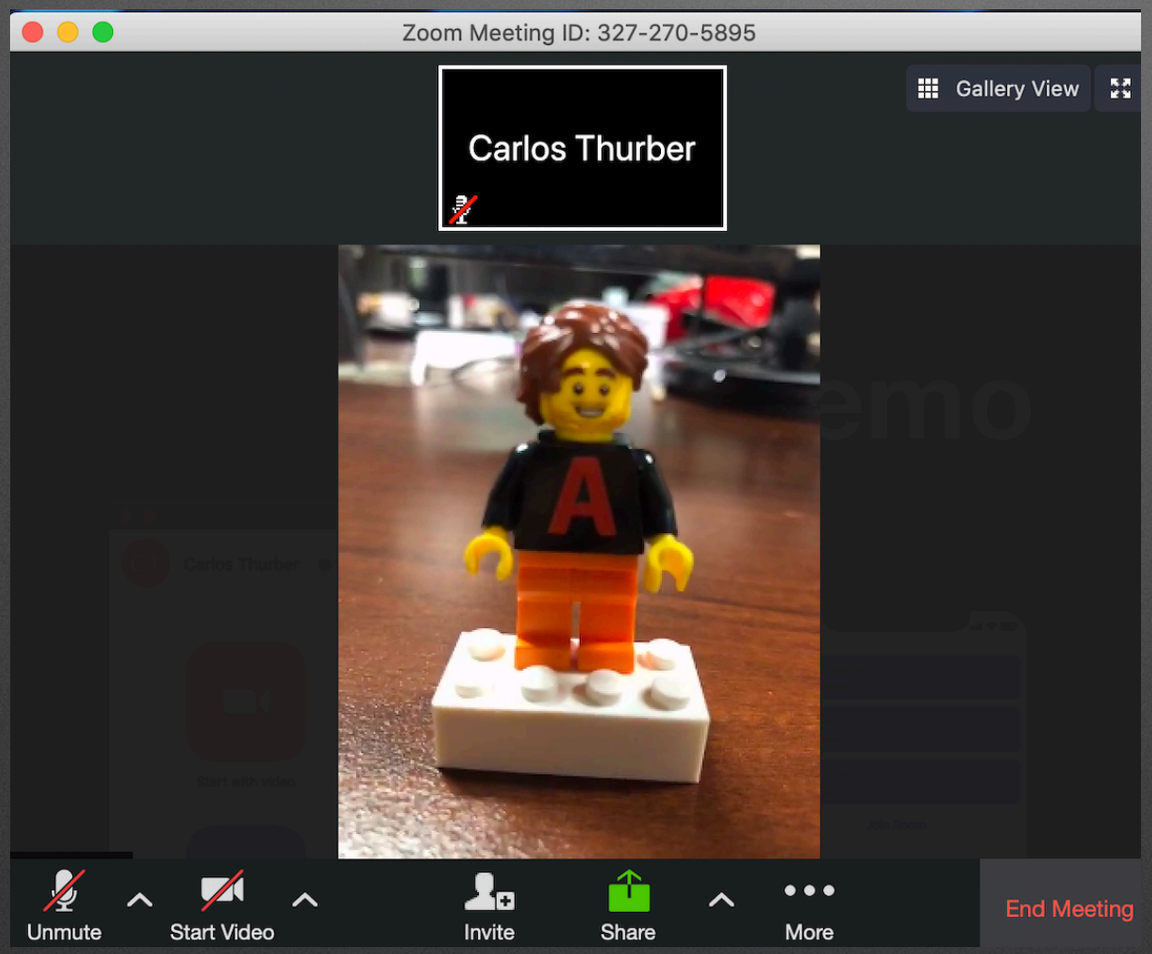

11:16
Meeting Number
Password
Display Name
Carlos-iOS
Join Room

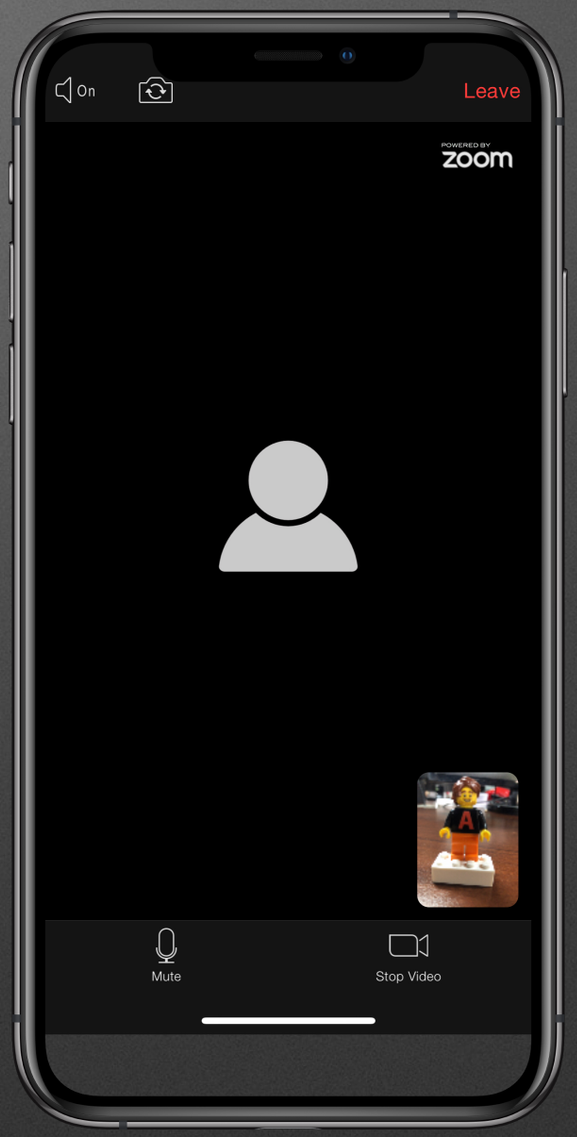
Demo


Zoom Meeting ID: 327-270-5895
Gallery View
Unmute
Start Video
Invite
Share
More
End Meeting

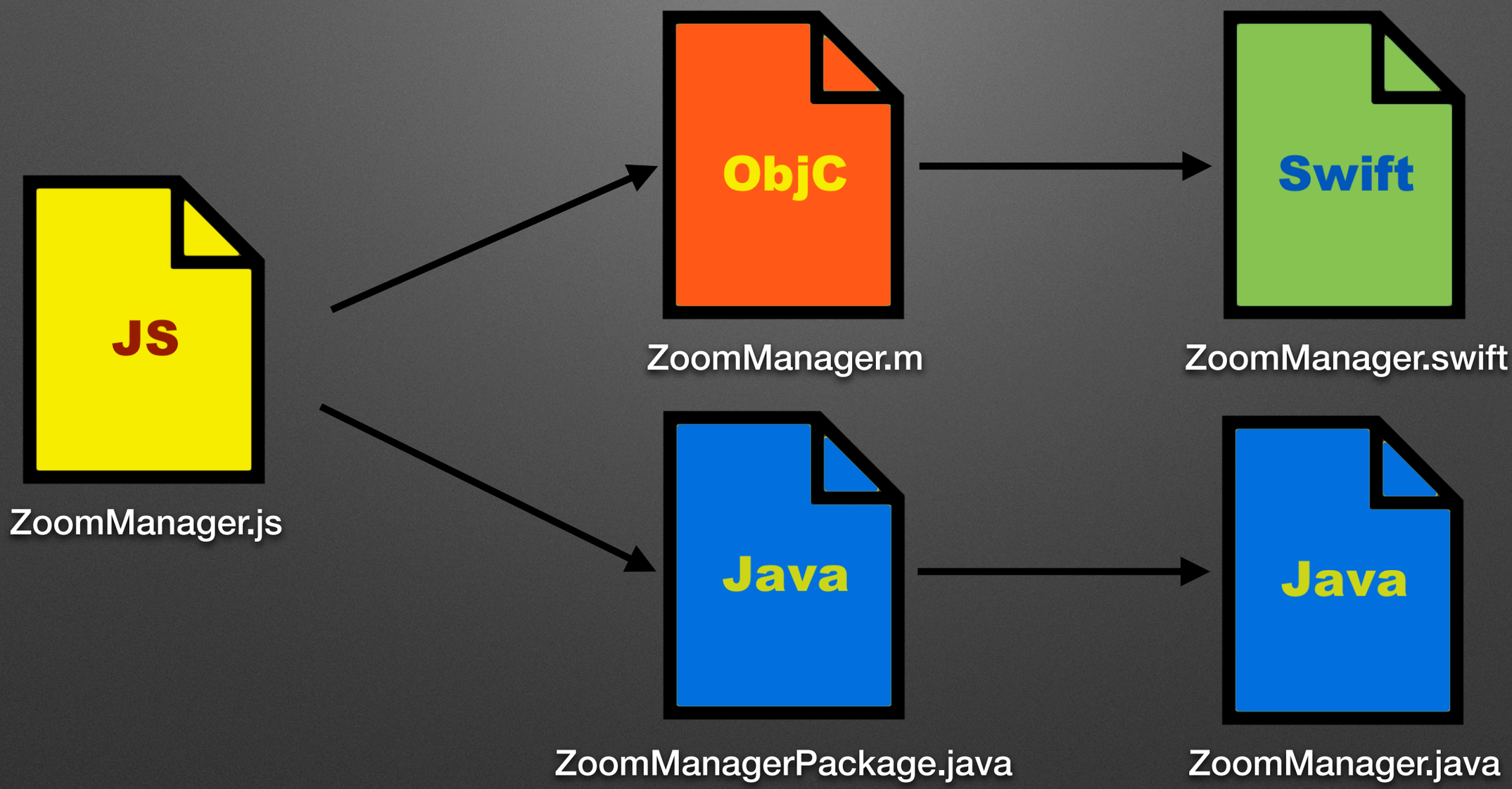

10:28
Meeting Number
3272705895
Password
••••••••
Display Name
Carlos-iOS
Join Room

Demo


Zoom Meeting ID: 327-270-5895
Carlos Thurber
Gallery View
Unmute
Start Video
Invite
Share
More
End Meeting


On
Leave
POWERED BY
zoom
Mute
Stop Video

Module Components

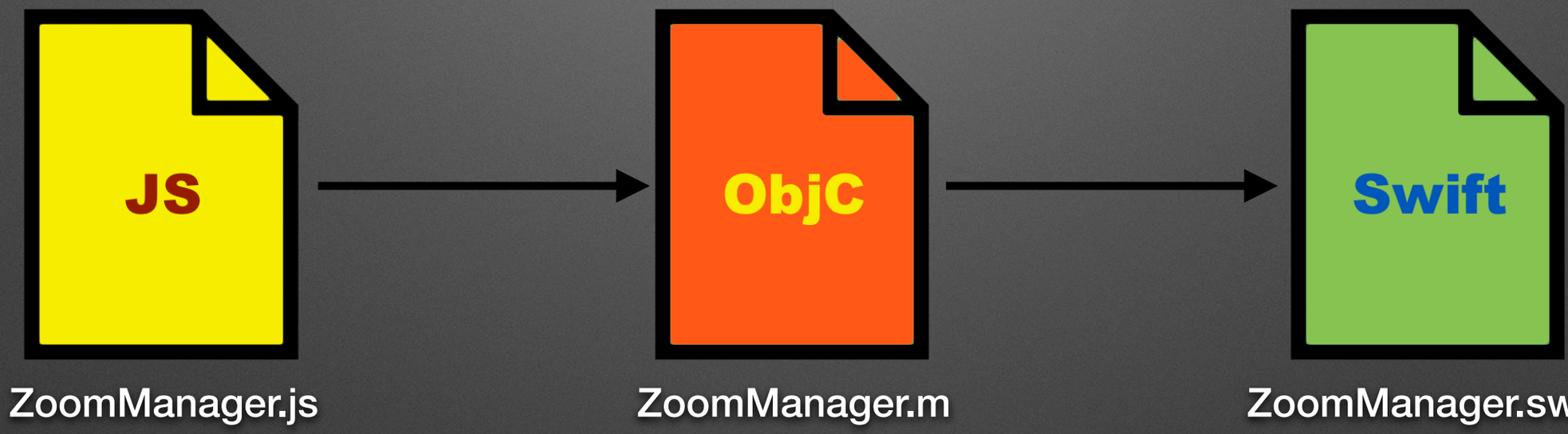


JS Wrapper

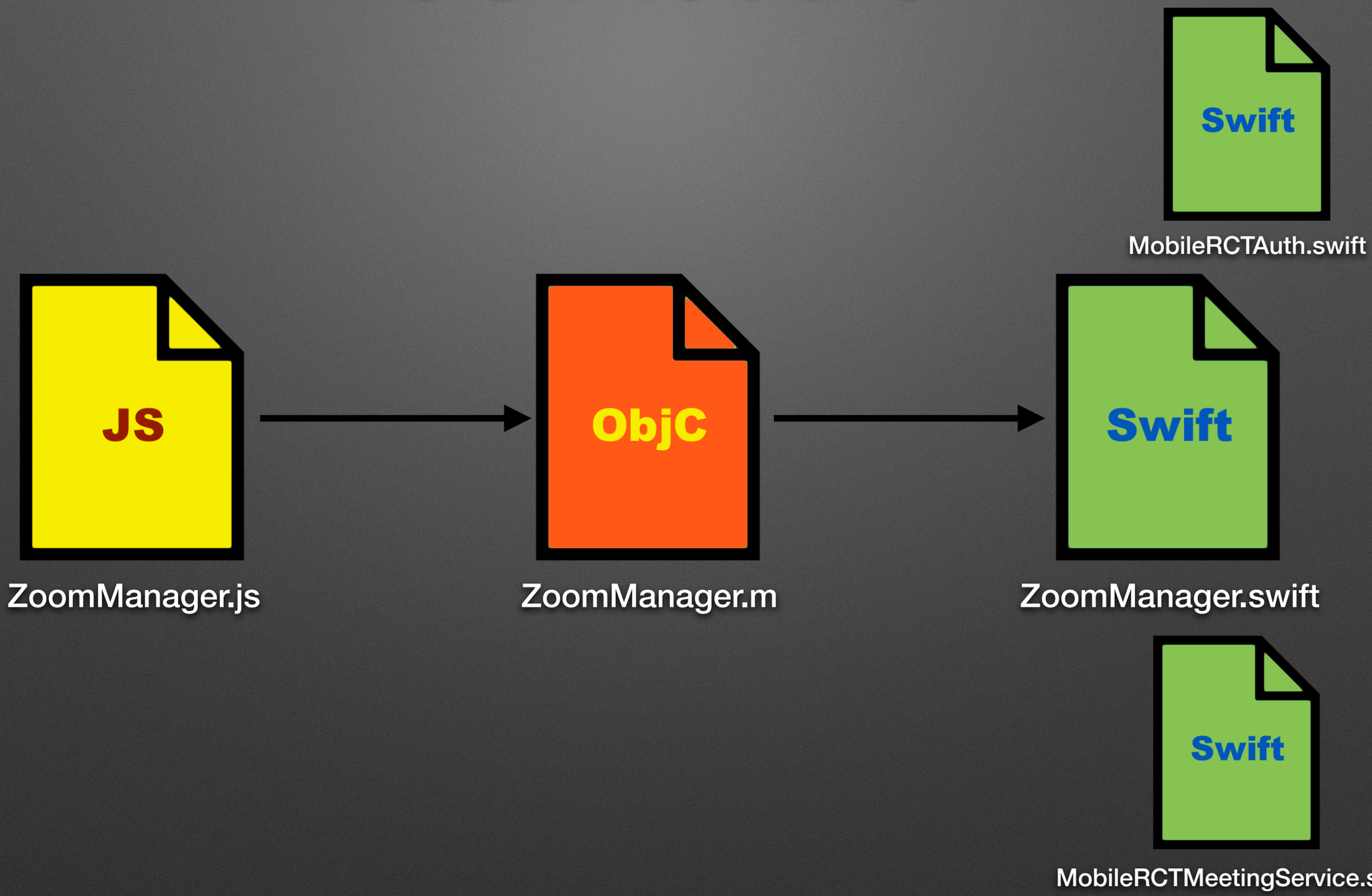
ZoomManager.js

```
import { Alert, NativeModules } from 'react-native';
import config from '../config';

const initialize = async () => {
  try {
    await NativeModules.ZoomManager.initialize(
      config['ZOOM_SDK_KEY'],
      config['ZOOM_SDK_SECRET']
    );
    Alert.alert('Zoom SDK Initialized');
  } catch (error) {
    Alert.alert('Failed Initialization', error.message);
  }
};

const joinMeeting = async (meetingNumber, userName, pwd) => {
  try {
    await NativeModules.ZoomManager.joinMeeting({
      meetingNumber,
      userName,
      pwd
    });
  } catch (error) {
    Alert.alert('Failed joining to room', error.message);
  }
};

module.exports = {
  initialize,
  joinMeeting
};
```

iOS Module


JS
ObjC
Swift
ZoomManager.js
ZoomManager.m
ZoomManager.swift

iOS Module


Swift
MobileRCTAuth.swift
JS
ObjC
Swift
ZoomManager.js
ZoomManager.m
ZoomManager.swift
Swift
MobileRCTMeetingService.swift

iOS Module

ZoomManager.m

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

@interface RCT_EXTERN_MODULE(RCTZoomManager, NSObject)

RCT_EXTERN_METHOD(
                  initialize:(NSString*)key
                  secret:(NSString*)secret
                  resolver:(RCTPromiseResolveBlock)resolver
                  rejecter:(RCTPromiseRejectBlock)rejecter
                  )

RCT_EXTERN_METHOD(
                  joinMeeting:(NSDictionary *)options
                  resolver:(RCTPromiseResolveBlock)resolver
                  rejecter:(RCTPromiseRejectBlock)rejecter
                  )

@end
```

iOS Module

ZoomManager.swift

```swift
@objc (RCTZoomManager)
class RCTZoomManager: NSObject {

  @objc var methodQueue = DispatchQueue.main

  @objc static func requiresMainQueueSetup() -> Bool {
    return true
  }

  var promiseResolver : RCTPromiseResolveBlock?
  var promiseRejecter : RCTPromiseRejectBlock?

}
```

iOS Module: init

MobileRTCAuth.swift

```
@objc
extension RCTZoomManager : MobileRTCAuthDelegate {

  @objc func initialize(_
    key:String,
    secret:String,
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {

  }
```

iOS Module: init

MobileRTCAuth.swift

```
@objc
extension RCTZoomManager : MobileRTCAuthDelegate {

  @objc func initialize(_
    key:String,
    secret:String,
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {

    guard let mobileRTC = MobileRTC.shared(),
          let authService = mobileRTC.getAuthService()
    else {
      rejecter("-1", "Failed to initialize", nil)
      return
     }

    self.promiseResolver = resolver
    self.promiseRejecter = rejecter



}
```

iOS Module: init

MobileRTCAuth.swift

```
@objc
extension RCTZoomManager : MobileRTCAuthDelegate {

  @objc func initialize(_
    key:String,
    secret:String,
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {
    mobileRTC.setMobileRTCDomain("zoom.us")
    authService.delegate = self
    authService.clientKey = key
    authService.clientSecret = secret
    authService.sdkAuth()
  }
```

iOS Module: init

MobileRTCAuth.swift

```swift
@objc
extension RCTZoomManager : MobileRTCAuthDelegate {

  func onMobileRTCAuthReturn(_
    returnValue: MobileRTCAuthError
  ) {

    guard let promiseResolver = self.promiseResolver,
          let promiseRejecter = self.promiseRejecter else { return }

    if returnValue == MobileRTCAuthError_Success {
      promiseResolver(nil)
    }
    else {
      let errorCode = Int(returnValue.rawValue)
      promiseRejecter("\(errorCode)", "Failed to initialize", nil)
    }

    self.promiseResolver = nil
    self.promiseRejecter = nil
  }

}
```

iOS Module: Join

MobileRTCMeetingService.swift

```swift
@objc
extension RCTZoomManager : MobileRTCMeetingServiceDelegate {

  @objc func joinMeeting(_
    options:[String:Any],
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {









}
```

iOS Module: Join

MobileRTCMeetingService.swift

```swift
@objc
extension RCTZoomManager : MobileRTCMeetingServiceDelegate {

  @objc func joinMeeting(_
    options:[String:Any],
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {

    guard let mobileRTC = MobileRTC.shared(),
          let meetingService = mobileRTC.getMeetingService(),
          let meetingSettings = mobileRTC.getMeetingSettings()
    else {
      rejecter("-1", "Failed to initialize", nil)
      return
    }

    self.promiseResolver = resolver
    self.promiseRejecter = rejecter




  }
```

iOS Module: Join

MobileRTCMeetingService.swift

```swift
@objc
extension RCTZoomManager : MobileRTCMeetingServiceDelegate {

  @objc func joinMeeting(_
    options:[String:Any],
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {

    meetingSettings.meetingTitleHidden = true
    meetingSettings.meetingShareHidden = true
    meetingSettings.meetingParticipantHidden = true
    meetingSettings.meetingMoreHidden = true
    meetingSettings.setAutoConnectInternetAudio(true)

  }
```

iOS Module: Join

MobileRTCMeetingService.swift

```
@objc
extension RCTZoomManager : MobileRTCMeetingServiceDelegate {

  @objc func joinMeeting(_
    options:[String:Any],
    resolver:@escaping RCTPromiseResolveBlock,
    rejecter:@escaping RCTPromiseRejectBlock
  ) {
```

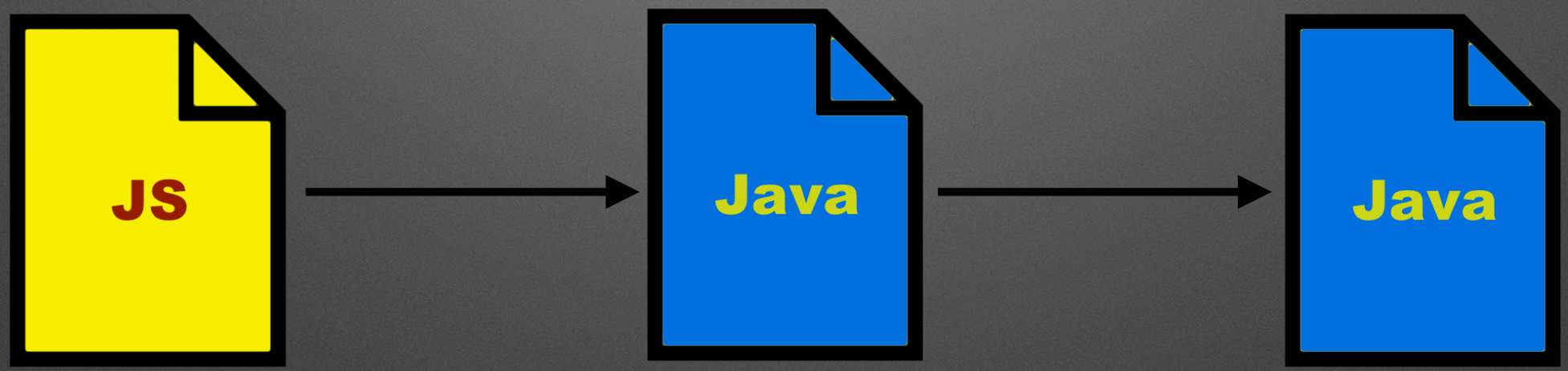
```
    meetingService.delegate = self
    let paramsDict : [AnyHashable : Any] = [
      kMeetingParam_Username:options["userName"] ?? "",
      kMeetingParam_MeetingNumber:options["meetingNumber"] ?? "",
      kMeetingParam_MeetingPassword:options["pwd"] ?? ""
    ]

    let ret = meetingService.joinMeeting(with: paramsDict)
  }
```

iOS Module: Join

MobileRTCMeetingService.swift

```swift
@objc
extension RCTZoomManager : MobileRTCMeetingServiceDelegate {

  func onMeetingError(_
    error: MobileRTCMeetError,
    message: String!
  ) {
    guard let promiseResolver = self.promiseResolver,
          let promiseRejecter = self.promiseRejecter else { return }

    if error == MobileRTCMeetError_Success {
      promiseResolver(nil)
    }
    else {
      let errorCode = Int(error.rawValue)
      promiseRejecter("\(errorCode)", message, nil)
    }

    self.promiseResolver = nil
    self.promiseRejecter = nil
  }

}
```

Android Module



ZoomManager.js

ZoomManagerPackage.java

ZoomManager.java

Android Module

ZoomManagerPackage.java

```java
public class ZoomManagerPackage implements ReactPackage {

    @Override
    public List<ViewManager> createViewManagers(ReactApplicationContext reactContext) {
        return Collections.emptyList();
    }

    @Override
    public List<NativeModule> createNativeModules(ReactApplicationContext reactContext) {
        List<NativeModule> modules = new ArrayList<>();
        modules.add(new ZoomManager(reactContext));
        return modules;
    }

}
```

Android Module

MainApplication.java

```java
public class MainApplication extends Application implements ReactApplication {

    @Override
    protected List<ReactPackage> getPackages() {
      return Arrays.<ReactPackage>asList(
          new MainReactPackage(),
          new ZoomManagerPackage()
      );
    }

}
```

Android Module

ZoomManager.java

```
public class ZoomManager extends ReactContextBaseJavaModule
                                                        {




    public ZoomManager(ReactApplicationContext reactContext) {
        super(reactContext);
    }

    @Override
    public String getName() {
        return "RCTZoomManager";
    }
```

Android Module

ZoomManager.java

```
public class ZoomManager extends ReactContextBaseJavaModule implements
  ZoomSDKInitializeListener, MeetingServiceListener {




    public ZoomManager(ReactApplicationContext reactContext) {
        super(reactContext);
```

```
}

@Override
public String getName() {
    return "RCTZoomManager";
}
```

Android Module

ZoomManager.java

```
public class ZoomManager extends ReactContextBaseJavaModule implements
  ZoomSDKInitializeListener, MeetingServiceListener {


    private final JoinMeetingOptions joinMeetingOptions;

    public ZoomManager(ReactApplicationContext reactContext) {
        super(reactContext);
        joinMeetingOptions = new JoinMeetingOptions();
        joinMeetingOptions.no_disconnect_audio = true;
        joinMeetingOptions.meeting_views_options =
          this.joinMeetingOptions.meeting_views_options
                | MeetingViewsOptions.NO_TEXT_MEETING_ID
                | MeetingViewsOptions.NO_TEXT_PASSWORD
                | MeetingViewsOptions.NO_BUTTON_SHARE
                | MeetingViewsOptions.NO_BUTTON_PARTICIPANTS
                | MeetingViewsOptions.NO_BUTTON_MORE;
    }

    @Override
    public String getName() {
        return "RCTZoomManager";
    }
```

Android Module

ZoomManager.java

```java
public class ZoomManager extends ReactContextBaseJavaModule implements
  ZoomSDKInitializeListener, MeetingServiceListener {

    private Promise mPromise;
    private final JoinMeetingOptions joinMeetingOptions;

    public ZoomManager(ReactApplicationContext reactContext) {
        super(reactContext);

        joinMeetingOptions = new JoinMeetingOptions();
        joinMeetingOptions.no_disconnect_audio = true;
        joinMeetingOptions.meeting_views_options =
          this.joinMeetingOptions.meeting_views_options
                | MeetingViewsOptions.NO_TEXT_MEETING_ID
                | MeetingViewsOptions.NO_TEXT_PASSWORD
                | MeetingViewsOptions.NO_BUTTON_SHARE
                | MeetingViewsOptions.NO_BUTTON_PARTICIPANTS
                | MeetingViewsOptions.NO_BUTTON_MORE;
    }

    @Override
    public String getName() {
        return "RCTZoomManager";
    }
```

Android Module: init

ZoomManager.java

```java
@ReactMethod
public void initialize(final String sdkKey, final String sdkSecret, final Promise promise) {




}
```

Android Module: init

ZoomManager.java

```java
@ReactMethod
public void initialize(final String sdkKey, final String sdkSecret, final Promise promise) {
    this.getCurrentActivity().runOnUiThread(new Runnable() {
        public void run() {
            initSDK(sdkKey, sdkSecret, "zoom.us", promise);
        }
    });
}
```

Android Module: init

ZoomManager.java

```
@ReactMethod
public void initialize(final String sdkKey, final String sdkSecret, final Promise promise) {
    this.getCurrentActivity().runOnUiThread(new Runnable() {
        public void run() {
            initSDK(sdkKey, sdkSecret, "zoom.us", promise);
        }
    });
}

private void initSDK(String sdkKey, String sdkSecret, String sdkDomain, Promise promise) {
    mPromise = promise;


}
```

Android Module: init

ZoomManager.java

```
@ReactMethod
public void initialize(final String sdkKey, final String sdkSecret, final Promise promise) {
    this.getCurrentActivity().runOnUiThread(new Runnable() {
        public void run() {
            initSDK(sdkKey, sdkSecret, "zoom.us", promise);
        }
    });
}

private void initSDK(String sdkKey, String sdkSecret, String sdkDomain, Promise promise) {
    mPromise = promise;

    ZoomSDK zoomSDK = ZoomSDK.getInstance();
    zoomSDK.initialize(this.getCurrentActivity(), sdkKey, sdkSecret, sdkDomain, this);
}
```

Android Module: init

ZoomManager.java

```
@ReactMethod
public void initialize(final String sdkKey, final String sdkSecret, final Promise promise) {
    this.getCurrentActivity().runOnUiThread(new Runnable() {
        public void run() {
            initSDK(sdkKey, sdkSecret, "zoom.us", promise);
        }
    });
}

private void initSDK(String sdkKey, String sdkSecret, String sdkDomain, Promise promise) {
    mPromise = promise;

    ZoomSDK zoomSDK = ZoomSDK.getInstance();
    zoomSDK.initialize(this.getCurrentActivity(), sdkKey, sdkSecret, sdkDomain, this);
}

// ZoomSDKInitializeListener

@Override
public void onZoomSDKInitializeResult(int errorCode, int internalErrorCode) {
    if (errorCode == ZoomError.ZOOM_ERROR_SUCCESS) {
        mPromise.resolve("Success!");
    } else {
        mPromise.reject(""+errorCode, "Failed to initialize");
    }
    mPromise = null;
}
```

Android Module: Join

ZoomManager.java

```java
@ReactMethod
public void joinMeeting(ReadableMap options, Promise promise) {

}
```

Android Module: Join

ZoomManager.java

```java
@ReactMethod
public void joinMeeting(ReadableMap options, Promise promise) {
    mPromise = promise;
}
```

Android Module: Join

ZoomManager.java

```java
@ReactMethod
public void joinMeeting(ReadableMap options, Promise promise) {
    mPromise = promise;

    JoinMeetingParams meetingParams = new JoinMeetingParams();
    meetingParams.meetingNo = options.getString("meetingNumber");
    meetingParams.displayName = options.getString("userName");
    meetingParams.password = options.getString("pwd");

}
```

Android Module: Join

ZoomManager.java

```java
@ReactMethod
public void joinMeeting(ReadableMap options, Promise promise) {
    mPromise = promise;

    JoinMeetingParams meetingParams = new JoinMeetingParams();
    meetingParams.meetingNo = options.getString("meetingNumber");
    meetingParams.displayName = options.getString("userName");
    meetingParams.password = options.getString("pwd");

    ZoomSDK zoomSDK = ZoomSDK.getInstance();
    MeetingService meetingService = zoomSDK.getMeetingService();
    meetingService.addListener(this);
    meetingService.joinMeetingWithParams(this.getCurrentActivity(), meetingParams, this.joinMeetingOptions);
}
```

Android Module: Join

ZoomManager.java

```java
@ReactMethod
public void joinMeeting(ReadableMap options, Promise promise) {
    mPromise = promise;

    JoinMeetingParams meetingParams = new JoinMeetingParams();
    meetingParams.meetingNo = options.getString("meetingNumber");
    meetingParams.displayName = options.getString("userName");
    meetingParams.password = options.getString("pwd");

    ZoomSDK zoomSDK = ZoomSDK.getInstance();
    MeetingService meetingService = zoomSDK.getMeetingService();
    meetingService.addListener(this);
    meetingService.joinMeetingWithParams(this.getCurrentActivity(), meetingParams, this.joinMeetingOptions);
}

// MeetingServiceListener

@Override
public void onMeetingStatusChanged(MeetingStatus meetingStatus, int errorCode, int internalErrorCode) {

    if (mPromise == null) {
        return;
    }

    if (meetingStatus == MeetingStatus.MEETING_STATUS_INMEETING) {
        mPromise.resolve("Success!");
        mPromise = null;
    }
    else if (meetingStatus == MeetingStatus.MEETING_STATUS_FAILED) {
        mPromise.reject(""+errorCode, "Meeting Failed");
        mPromise = null;
    }
}
```

https://github.com/calitb/RN-NativeModules

Thanks!!