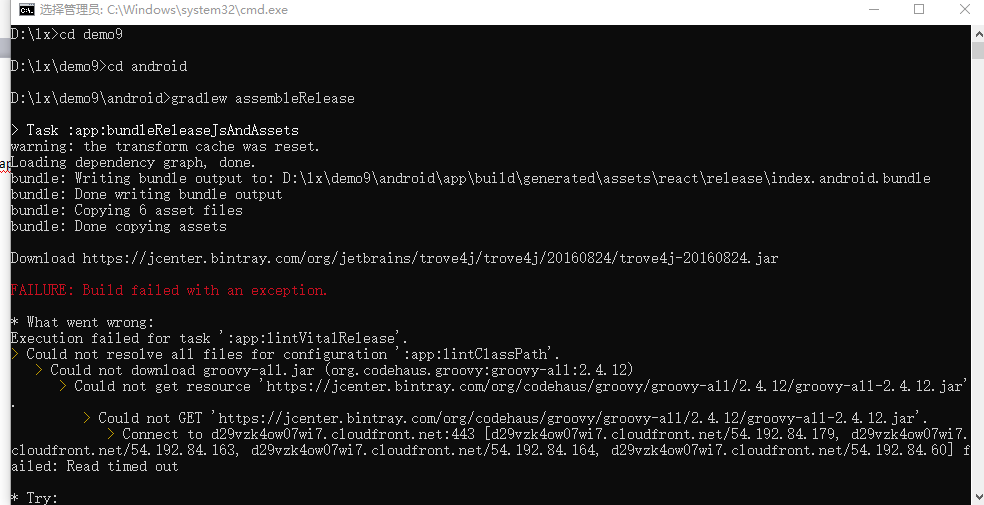
React native apk 打包



报错，解决方案：

AILURE: Build failed with an exception.

\* What went wrong:

Execution failed for task ':app:lintVitalRelease

解决方案：

<https://blog.csdn.net/swallow_csdn/article/details/83012642>

打包时的错误提示：

Error:Execution failed for task ':app:lintVitalRelease'. > java.lang.IllegalStateException: Expected a name but was STRING at line 1 column 99 path $[0].apkInfo.versionName

debug版可以运行，release版生成不了。似乎是有没有用到的module,只需在app下的gradle文件的android部分添加代码：

lintOptions {

checkReleaseBuilds false

abortOnError false

Android/app/build.gradle文件

apply plugin: "com.android.application"

import com.android.build.OutputFile

/\*\*

\* The react.gradle file registers a task for each build variant (e.g. bundleDebugJsAndAssets

\* and bundleReleaseJsAndAssets).

\* These basically call `react-native bundle` with the correct arguments during the Android build

\* cycle. By default, bundleDebugJsAndAssets is skipped, as in debug/dev mode we prefer to load the

\* bundle directly from the development server. Below you can see all the possible configurations

\* and their defaults. If you decide to add a configuration block, make sure to add it before the

\* `apply from: "../../node\_modules/react-native/react.gradle"` line.

\*

\* project.ext.react = [

\* // the name of the generated asset file containing your JS bundle

\* bundleAssetName: "index.android.bundle",

\*

\* // the entry file for bundle generation

\* entryFile: "index.android.js",

\*

\* // whether to bundle JS and assets in debug mode

\* bundleInDebug: false,

\*

\* // whether to bundle JS and assets in release mode

\* bundleInRelease: true,

\*

\* // whether to bundle JS and assets in another build variant (if configured).

\* // See http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Build-Variants

\* // The configuration property can be in the following formats

\* // 'bundleIn${productFlavor}${buildType}'

\* // 'bundleIn${buildType}'

\* // bundleInFreeDebug: true,

\* // bundleInPaidRelease: true,

\* // bundleInBeta: true,

\*

\* // whether to disable dev mode in custom build variants (by default only disabled in release)

\* // for example: to disable dev mode in the staging build type (if configured)

\* devDisabledInStaging: true,

\* // The configuration property can be in the following formats

\* // 'devDisabledIn${productFlavor}${buildType}'

\* // 'devDisabledIn${buildType}'

\*

\* // the root of your project, i.e. where "package.json" lives

\* root: "../../",

\*

\* // where to put the JS bundle asset in debug mode

\* jsBundleDirDebug: "$buildDir/intermediates/assets/debug",

\*

\* // where to put the JS bundle asset in release mode

\* jsBundleDirRelease: "$buildDir/intermediates/assets/release",

\*

\* // where to put drawable resources / React Native assets, e.g. the ones you use via

\* // require('./image.png')), in debug mode

\* resourcesDirDebug: "$buildDir/intermediates/res/merged/debug",

\*

\* // where to put drawable resources / React Native assets, e.g. the ones you use via

\* // require('./image.png')), in release mode

\* resourcesDirRelease: "$buildDir/intermediates/res/merged/release",

\*

\* // by default the gradle tasks are skipped if none of the JS files or assets change; this means

\* // that we don't look at files in android/ or ios/ to determine whether the tasks are up to

\* // date; if you have any other folders that you want to ignore for performance reasons (gradle

\* // indexes the entire tree), add them here. Alternatively, if you have JS files in android/

\* // for example, you might want to remove it from here.

\* inputExcludes: ["android/\*\*", "ios/\*\*"],

\*

\* // override which node gets called and with what additional arguments

\* nodeExecutableAndArgs: ["node"],

\*

\* // supply additional arguments to the packager

\* extraPackagerArgs: []

\* ]

\*/

project.ext.react = [

entryFile: "index.js"

]

apply from: "../../node\_modules/react-native/react.gradle"

/\*\*

\* Set this to true to create two separate APKs instead of one:

\* - An APK that only works on ARM devices

\* - An APK that only works on x86 devices

\* The advantage is the size of the APK is reduced by about 4MB.

\* Upload all the APKs to the Play Store and people will download

\* the correct one based on the CPU architecture of their device.

\*/

def enableSeparateBuildPerCPUArchitecture = false

/\*\*

\* Run Proguard to shrink the Java bytecode in release builds.

\*/

def enableProguardInReleaseBuilds = false

android {

compileSdkVersion rootProject.ext.compileSdkVersion

buildToolsVersion rootProject.ext.buildToolsVersion

defaultConfig {

applicationId "com.demo9"

minSdkVersion rootProject.ext.minSdkVersion

targetSdkVersion rootProject.ext.targetSdkVersion

versionCode 1

versionName "1.0"

ndk {

abiFilters "armeabi-v7a", "x86"

}

}

signingConfigs {

release {

storeFile file("D:/lx/demo9/android/app/my-release-key.keystore")

storePassword "123456"

keyAlias "my-key-alias"

keyPassword "123456"

}

}

splits {

abi {

reset()

enable enableSeparateBuildPerCPUArchitecture

universalApk false // If true, also generate a universal APK

include "armeabi-v7a", "x86"

}

}

buildTypes {

release {

minifyEnabled enableProguardInReleaseBuilds

proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"

signingConfig signingConfigs.release

}

}

// applicationVariants are e.g. debug, release

applicationVariants.all { variant ->

variant.outputs.each { output ->

// For each separate APK per architecture, set a unique version code as described here:

// http://tools.android.com/tech-docs/new-build-system/user-guide/apk-splits

def versionCodes = ["armeabi-v7a":1, "x86":2]

def abi = output.getFilter(OutputFile.ABI)

if (abi != null) { // null for the universal-debug, universal-release variants

output.versionCodeOverride =

versionCodes.get(abi) \* 1048576 + defaultConfig.versionCode

}

}

}

lintOptions {

checkReleaseBuilds false

abortOnError false

}

}

dependencies {

implementation fileTree(dir: "libs", include: ["\*.jar"])

implementation "com.android.support:appcompat-v7:${rootProject.ext.supportLibVersion}"

implementation "com.facebook.react:react-native:+" // From node\_modules

}

// Run this once to be able to run the application with BUCK

// puts all compile dependencies into folder libs for BUCK to use

task copyDownloadableDepsToLibs(type: Copy) {

from configurations.compile

into 'libs'

}

