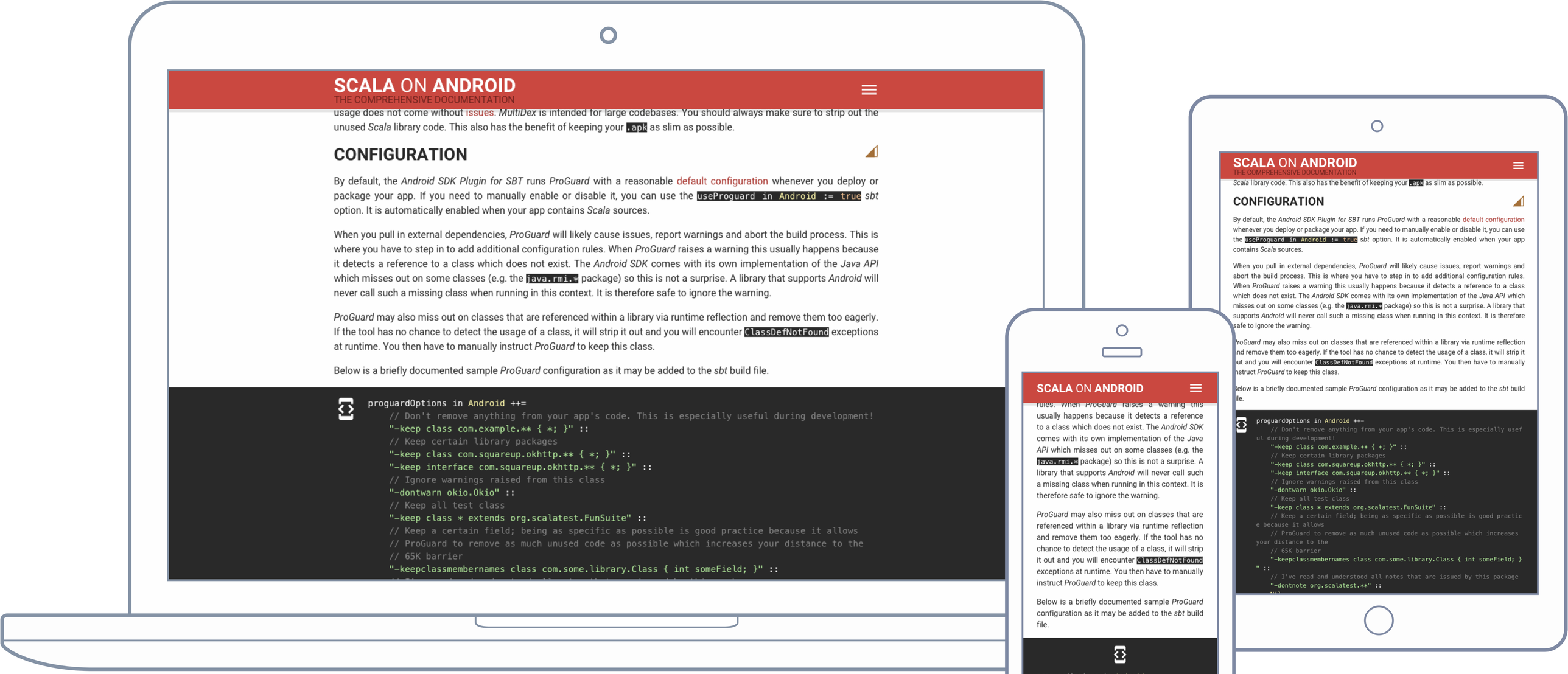
SCALA ON ANDROID
THE COMPREHENSIVE DOCUMENTATION
usage does not come without issues. MultiDex is intended for large codebases. You should always make sure to strip out the unused Scala library code. This also has the benefit of keeping your .apk as slim as possible.
CONFIGURATION
By default, the Android SDK Plugin for SBT runs ProGuard with a reasonable default configuration whenever you deploy or package your app. If you need to manually enable or disable it, you can use the useProguard in Android := true sbt option. It is automatically enabled when your app contains Scala sources.
When you pull in external dependencies, ProGuard will likely cause issues, report warnings and abort the build process. This is where you have to step in to add additional configuration rules. When ProGuard raises a warning this usually happens because it detects a reference to a class which does not exist. The Android SDK comes with its own implementation of the Java API which misses out on some classes (e.g. the java.rmi.* package) so this is not a surprise. A library that supports Android will never call such a missing class when running in this context. It is therefore safe to ignore the warning.
ProGuard may also miss out on classes that are referenced within a library via runtime reflection and remove them too eagerly. If the tool has no chance to detect the usage of a class, it will strip it out and you will encounter ClassDefNotFound exceptions at runtime. You then have to manually instruct ProGuard to keep this class.
Below is a briefly documented sample ProGuard configuration as it may be added to the sbt build file.
proguardOptions in Android ++=
// Don't remove anything from your app's code. This is especially useful during development!
"-keep class com.example.** { *; }" ::
// Keep certain library packages
"-keep class com.squareup.okhttp.** { *; }" ::
"-keep interface com.squareup.okhttp.** { *; }" ::
// Ignore warnings raised from this class
"-dontwarn okio.Okio" ::
// Keep all test class
"-keep class * extends org.scalatest.FunSuite" ::
// Keep a certain field; being as specific as possible is good practice because it allows
// ProGuard to remove as much unused code as possible which increases your distance to the
// 65K barrier
"-keepclassmembernames class com.some.library.Class { int someField; }" ::

SCALA ON ANDROID
THE COMPREHENSIVE DOCUMENTATION
Scala library code. This also has the benefit of keeping your .apk as slim as possible.
CONFIGURATION
By default, the Android SDK Plugin for SBT runs ProGuard with a reasonable default configuration whenever you deploy or package your app. If you need to manually enable or disable it, you can use the useProguard in Android := true sbt option. It is automatically enabled when your app contains Scala sources.
When you pull in external dependencies, ProGuard will likely cause issues, report warnings and abort the build process. This is where you have to step in to add additional configuration rules. When ProGuard raises a warning this usually happens because it detects a reference to a class which does not exist. The Android SDK comes with its own implementation of the Java API which misses out on some classes (e.g. the java.rmi.* package) so this is not a surprise. A library that supports Android will never call such a missing class when running in this context. It is therefore safe to ignore the warning.
ProGuard may also miss out on classes that are referenced within a library via runtime reflection and remove them too eagerly. If the tool has no chance to detect the usage of a class, it will strip it out and you will encounter ClassDefNotFound exceptions at runtime. You then have to manually instruct ProGuard to keep this class.
Below is a briefly documented sample ProGuard configuration as it may be added to the sbt build file.
proguardOptions in Android ++=
// Don't remove anything from your app's code. This is especially useful during development!
"-keep class com.example.** { *; }" ::
// Keep certain library packages
"-keep class com.squareup.okhttp.** { *; }" ::
"-keep interface com.squareup.okhttp.** { *; }" ::
// Ignore warnings raised from this class
"-dontwarn okio.Okio" ::
// Keep all test class
"-keep class * extends org.scalatest.FunSuite" ::
// Keep a certain field; being as specific as possible is good practice because it allows
// ProGuard to remove as much unused code as possible which increases your distance to the
// 65K barrier
"-keepclassmembernames class com.some.library.Class { int someField; }" ::
// I've read and understood all notes that are issued by this package
"-dontnote org.scalatest.**" ::

SCALA ON ANDROID
rules. When ProGuard raises a warning this usually happens because it detects a reference to a class which does not exist. The Android SDK comes with its own implementation of the Java API which misses out on some classes (e.g. the java.rmi.* package) so this is not a surprise. A library that supports Android will never call such a missing class when running in this context. It is therefore safe to ignore the warning.
ProGuard may also miss out on classes that are referenced within a library via runtime reflection and remove them too eagerly. If the tool has no chance to detect the usage of a class, it will strip it out and you will encounter ClassDefNotFound exceptions at runtime. You then have to manually instruct ProGuard to keep this class.
Below is a briefly documented sample ProGuard configuration as it may be added to the sbt build file.
proguardOptions in Android ++=
// Don't remove anything from your app's code. This is especially useful during development!
"-keep class com.example.** { *; }"