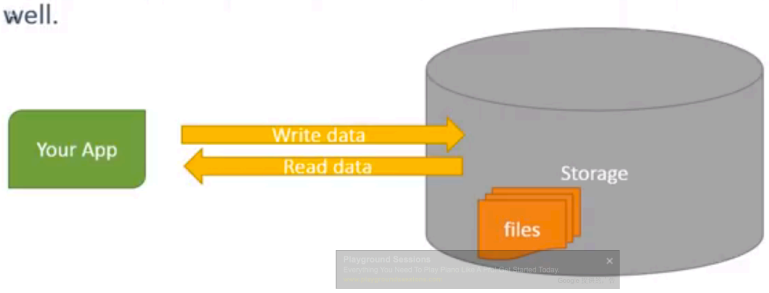
Internal Storage

- Internal allows you to read and write to files that are associated with each application's internal memory.
- These files can only be accessed by the application and cannot be accessed by other applications or users.
- When the application is uninstalled, these files are automatically removed as well.



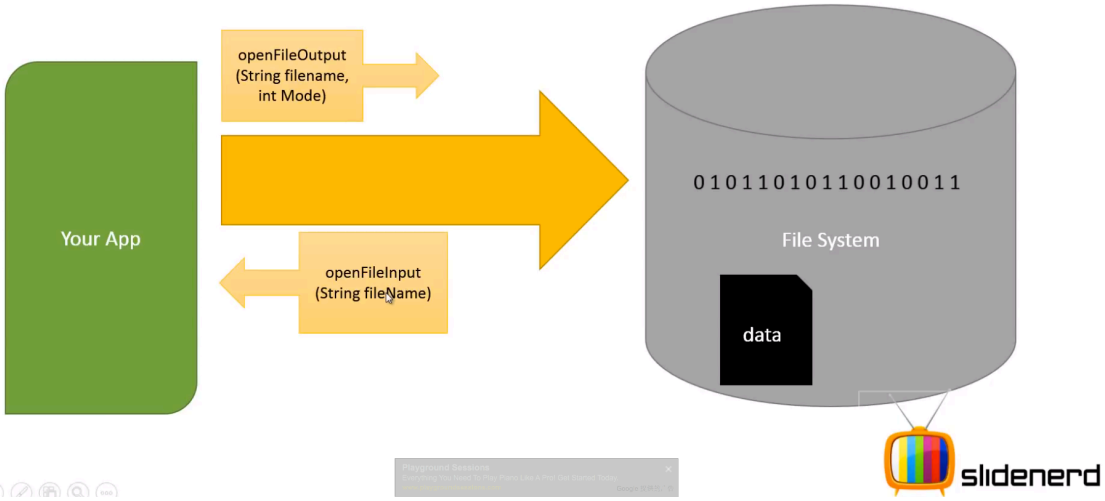
Internal Storage

- Your app's internal storage directory is specified by your app's package name in a special location of the Android file system.
- Other apps and users cannot browse your internal directories and do not have read or write access unless you explicitly set the files to be readable or writable.
- When you use MODE_PRIVATE for your files on the internal storage, they are never accessible to other apps.
- Internal storage is best when you want to be sure that neither the user nor other apps can access your files.

Context.MODE_PRIVATE: cannot be read by others

Context.MODE_APPEND: keep appending data to the existing contents

Internal Storage



When using MODE_PRIVATE, existing content with the same filename will be replaced.
When using MODE_APPEND, new content will be appended to the existing ones.
File storage works with Bytes. Before insert it to stream, we have to serialize it. Vise versa when we read data out, we have to de-serialize it.

Internal Storage

```
String FILENAME = “mytext.txt";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
byte[] buf=string.getBytes();
fos.write(buf);
fos.close();
```

Example that we will work out in java code

Our example