

# Saving to a JSON File in Jetpack Compose

Alex Reiffer

Github:

[https://github.com/HomeTownUS/  
JSONTutorial](https://github.com/HomeTownUS/JSONTutorial)

# Step 1: Add Dependencies

```
//In libs.versions.toml
```

```
kotlin-serialization = {id = "org.jetbrains.kotlin.plugin.serialization", version.ref = "kotlin"}
```

```
plugins { //In top-level build.gradle.kts
```

```
    alias(libs.plugins.android.application) apply false
```

```
    alias(libs.plugins.kotlin.android) apply false
```

```
    alias(libs.plugins.kotlin.compose) apply false
```

```
    alias(libs.plugins.kotlin.serialization) apply false
```

```
}
```

```
plugins { //In App build.gradle.kts
```

```
    alias(libs.plugins.kotlin.serialization)
```

```
}
```

```
dependencies {
```

```
    implementation("org.jetbrains.kotlinx:kotlinx-serialization-json:1.6.0")
```

```
}
```

## Step 2: Write Data Class

The data class holds the information to be saved

```
@Serializable  
  
data class MyData(  
    val name:  
    String  
)
```

## Step 3: Save with Context

```
fun save(context: Context, text: String){
    val fileName = "save.json"
    val data = MyData(text)
    val saveString = Json.encodeToString(data)
    try {
        File(context.filesDir, fileName).writeText(saveString)
    } catch (e: Exception) {println(e)/*Do nothing*/ }
}
```

```
val context = LocalContext.current
Button(onClick = {save(context,text)}) {
    Text("Click to save name")
}
```

## Step 4: Load with Context

Use `runBlocking` to ensure `initialText` is set before text

```
val context = LocalContext.current

Column(modifier = Modifier.padding(16.dp)){

    val initialText = runBlocking { read(context) }

    var text by remember { mutableStateOf(initialText) }

}
```