JSON

**JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

use EditorJsonUtility.

ToJson to serialize other engine types to JSON).

Instead of serializing them directly, the method will attempt to serialize their public instance fields, producing an empty object as a result. Similarly, passing an array to this method will not produce a JSON array containing each element, but an object containing the public fields of the array object itself (of which there are none). To serialize the actual content of an array or primitive type, it is necessary to wrap it in a class or struct.

MyClass myObject = new MyClass();

myObject.level = 1;

myObject.timeElapsed = 47.5f;

myObject.playerName = "Dr Charles Francis";

Then use the JsonUtility.ToJson method to serialize it (convert it) to the JSON format:

string json = JsonUtility.ToJson(myObject);

// json now contains: '{"level":1,"timeElapsed":47.5,"playerName":"Dr Charles Francis"}'

To convert the JSON back into an object, use JsonUtility.FromJson:

myObject = JsonUtility.FromJson<MyClass>(json);

This creates a new instance of MyClass and sets the values on it using the JSON data.

If the JSON data contains values that do not map to fields in MyClass, then the serializer ignores those values.

If the JSON data is missing values for fields in MyClass, then the serializer leaves the constructed

values for those fields in the returned object.

You can also deserialize JSON data over an existing object, which overwrites any existing data:

JsonUtility.FromJsonOverwrite(json, myObject);