Quick Debugger Reference

public static void QuickLog<T>(T msg)

Imitation of Debug.Log().

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

msg - *Data to be printed. It must be the same type as T.*

EXAMPLE

```
string someWord = "ABCDEFGHIJKL";
int someInt = 3314;
Debugger.QuickLog<string>(someWord);
Debugger.QuickLog<int>(someInt);
```

---

public static void Log(string separator, ps string[] strings)

Prints all values in an array of text. Good for debugging two or more variables in the same call.

separator - *Separator*

strings - *Data to be printed.*

EXAMPLE

```
Debugger.Log(" ", "This", "is", "an", "example");
```

---

public static void Array<T>(T[] array)

Prints all values in an array separated with the default separator.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

array - *Array of T to be printed.*

EXAMPLE

```
string[] words = new string[5] { "one", "two", "three", "four", "five" };
Debugger.Array<string>(words);
```

---

public static void Array<T>(string separator, T[] array)

Prints all values in an array.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

separator - *Separator*

array - *Array of T to be printed.*

EXAMPLE

```
string[] words = new string[5] { "one", "two", "three", "four", "five" };
Debugger.Array<string>("; ", words);
```

---

public static void Array2D<T>(T[,] array)

Prints all values in a two-dimensional array separated with the default separator.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

array - *Array of T to be printed.*

EXAMPLE

```
string[,] twoArray = new string[,]
{
{"dog", "cat"},
{"day", "night"},
{"white", "black"},
{"happy", "sad"},
{"peace", "war"}
};
Debugger.Array2D<string>(twoArray);
```

---

public static void Array2D<T>(string separator, T[,] array)

Prints all values in a two-dimensional array.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

separator - *Separator*

array - *Array of T to be printed.*

EXAMPLE

```
string[,] twoArray = new string[,]
{
{"dog", "cat"},
{"day", "night"},
{"white", "black"},
{"happy", "sad"},
{"peace", "war"}
};
Debugger.Array2D<string>("; ", twoArray);
```

---

public static void Array3D<T>(T[,,] array)

Prints all values in a three-dimensional array separated with the default separator.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

array - *Array of T to be printed.*

EXAMPLE

```
int[, ,] threeArray = new int[3, 5, 4];
threeArray[0, 0, 0] = 1;
threeArray[0, 1, 0] = 2;
threeArray[0, 2, 0] = 3;
threeArray[0, 3, 0] = 4;
threeArray[0, 4, 0] = 5;
threeArray[1, 1, 1] = 2;
threeArray[2, 2, 2] = 3;
threeArray[2, 2, 3] = 4;
Debugger.Array3D<int>(threeArray);
```

---

public static void Array3D<T>(string separator, T[,,] array)

Prints all values in a three-dimensional array.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

separator - *Separator*

array - *Array of T to be printed.*

EXAMPLE

```
int[, ,] threeArray = new int[3, 5, 4];
threeArray[0, 0, 0] = 1;
threeArray[0, 1, 0] = 2;
threeArray[0, 2, 0] = 3;
threeArray[0, 3, 0] = 4;
threeArray[0, 4, 0] = 5;
threeArray[1, 1, 1] = 2;
threeArray[2, 2, 2] = 3;
threeArray[2, 2, 3] = 4;
Debugger.Array3D<int>("; ", threeArray);
```

---

public static void List<T>(List<T> list)

Prints all values in a list separated with the default separator

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

list - *List of T to be printed.*

EXAMPLE

```
List<Vector3> vectorsList = new List<Vector3>();
vectorsList.Add(Vector3.back);
vectorsList.Add(Vector3.down);
vectorsList.Add(Vector3.forward);
Debugger.List<Vector3>(vectorsList);
```

---

public static void List<T>(string separator, List<T> list)

Prints all values in a list

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

separator - *Separator*

list - *List of T to be printed.*

EXAMPLE

```
List<Vector3> vectorsList = new List<Vector3>();
vectorsList.Add(Vector3.back);
vectorsList.Add(Vector3.down);
vectorsList.Add(Vector3.forward);
Debugger.List<Vector3>("; ", vectorsList);
```

---

public static void Dictionary(Dictionary dictionary)

Prints all keys and values in a dictionary separated with the default separator

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

K - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

dictionary - *Dictionary with key of T and values of K to be printed.*

EXAMPLE

```
Dictionary dict = new Dictionary();
dict.Add(0, "zero");
dict.Add(1, "one");
dict.Add(2, "two");
dict.Add(3, "three");
dict.Add(4, "four");
Debugger.Dictionary(dict);
```

---

public static void Dictionary(string separator, Dictionary dictionary)

Prints all keys and values in a dictionary.

T - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

K - *Generic type parameter. When you call it, it can be any type: string, int, Vector3 etc.*

separator - *Separator*

dictionary - *Dictionary with key of T and values of K to be printed.*

EXAMPLE

```
Dictionary dict = new Dictionary();
dict.Add(0, "zero");
dict.Add(1, "one");
dict.Add(2, "two");
dict.Add(3, "three");
dict.Add(4, "four");
Debugger.Dictionary("; ", dict);
```

---

public static void ExceptionDetails(Exception e)

Prints exception with details.

e - *Exception to be printed*