# Understanding Non-nullable Reference Types in C# 8


**Jason Roberts**
.NET DEVELOPER

@robertsjason dontcodetired.com

# Overview

An overview of C# 8 null features

Enable non-nullable reference types

Specifying a reference should be nullable

Non-nullable properties

Non-nullable method return values

Null-coalescing and null-conditional operators

The null-forgiving operator

Refactoring existing code

Considerations

# An Overview of C# 8.0 Null Features

Design intent

Design enforcement

Is the variable, parameter, field, return value, etc. supposed to allow null values?

Sometimes your intent is that a reference *can* represent nothing/no value (i.e. null).

Sometimes your intent is that a reference should *never* be nothing (i.e. always have a value).

| | | |
|---|---|---|
| | | |
| | | |

| | Nullable Reference Type <C# 8 | |
|---|---|---|
| | | |
| | | |

| | Nullable Reference Type <C# 8 | |
|---|---|---|
| Dereference | | |
| Assign null | | |

| | Nullable Reference Type <C# 8 | |
|---|---|---|
| Dereference | Yes (null check) | |
| Assign null | | |

| | Nullable Reference Type <C# 8 | |
|---|---|---|
| Dereference | Yes (null check) | |
| Assign null | Yes | |

| | Nullable Reference Type <C# 8 | Non-Nullable Reference Type C# 8 |
|---|---|---|
| Dereference | Yes (null check) | |
| Assign null | Yes | |

| | Nullable Reference Type <C# 8 | Non-Nullable Reference Type C# 8 |
|---|---|---|
| Dereference | Yes (null check) | Yes |
| Assign null | Yes | |

| | Nullable Reference Type <C# 8 | Non-Nullable Reference Type C# 8 |
|---|---|---|
| Dereference | Yes (null check) | Yes |
| Assign null | Yes | No |

Enforced by compiler

Examine developer's design intent

Possible intent violations:

- Warning
- Error

Opt-in

Existing code

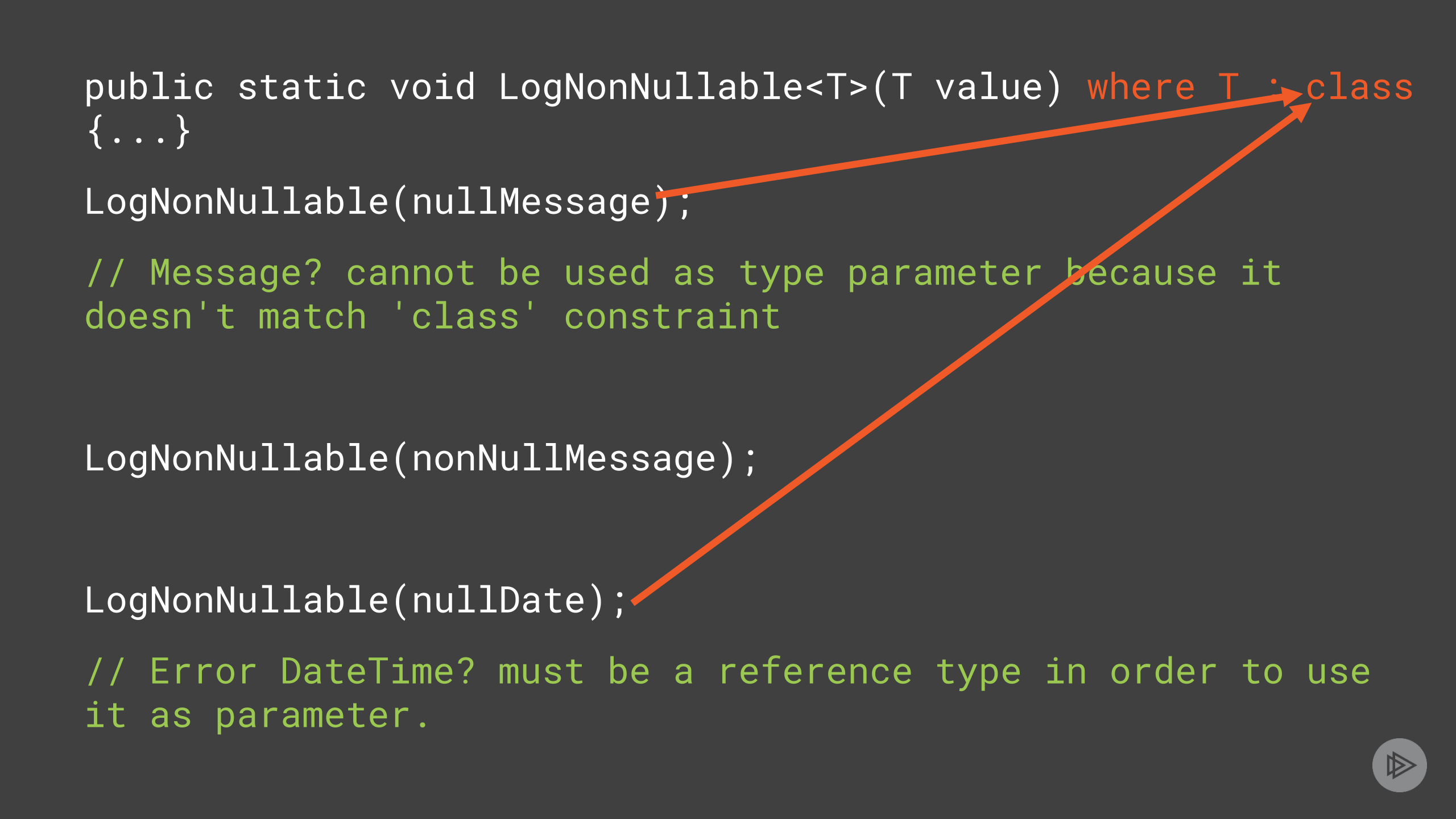
# Nullable and Non-nullable Generic Types

```
Message? nullMessage = null;
Message nonNullMessage = new Message();


List<Message> m1 = new List<Message>();
m1.Add(nullMessage); // Warning: Possible null reference
m1.Add(nonNullMessage);


List<Message?> m2 = new List<Message?>();
m2.Add(nullMessage);
m2.Add(nonNullMessage);
```

```
public static void LogNullable<T>(T value) {...}
LogNullable(nullMessage);
LogNullable(nonNullMessage);

DateTime? nullDate = null;
LogNullable(nullDate);
```

```
public static void LogNullable<T>(T value) where T : class?
{...}


LogNullable(nullMessage);

LogNullable(nonNullMessage);


LogNullable(nullDate);

// Error 'DateTime?' must be a reference type
```

```
public static void LogNonNullable<T>(T value) where T : class
{...}

LogNonNullable(nullMessage);

// Message? cannot be used as type parameter because it
doesn't match 'class' constraint


LogNonNullable(nonNullMessage);


LogNonNullable(nullDate);

// Error DateTime? must be a reference type in order to use
it as parameter.
```



# Considerations

**Don’t try to remove all null values from your code**

**Instead express your intent**

**Not completely null-safe**

- Reflection
- Calling code that was compiled with feature disabled
- Analysis is limited in some cases

**Null checking code for public libraries**

**Automated tests**

# Summary

C# 8 nulls: design intent & enforcement

#nullable enable & #nullable disable

<Nullable>enable</Nullable>

Treating nullable warnings as errors

? nullable operator, e.g. string?

Variables, properties, method returns

?? & ?.

Null-forgiving operator !

Refactored existing code

Generic constraints: class and class?

Considerations