[0480] Nullable Types

A nullable data type is a type that can be assigned the value of null.

Normally, ValueTypes - the stack-based types that we talked about on the previous page - can never have the value of null. Assigning null to a local variable that's a ValueType will result in a compiler error.

If you want however, you can change things so that any ValueType can be set to null. You do that by adding a question mark to the right side of the type declaration. Like this:

```
int notNullable = null;   // Compiler error!
int? nullable = null;   // It works!
```

Now, WHY IN THE WORLD WOULD YOU WANT TO DO THIS? That's a very good question. It turns out that data from databases can sometimes need this capability. I would argue that the database needs to be fixed, but whatever.

But wait... there's more.

The Null Coalescing Operator

The Null Coalescing Operator is a strange name for a simple concept. It allows you to set a value that should be used whenever the preceding expression is null. It is represented by two question marks put together.

```
int? myAge = AskMyAge() ?? 29;
```

That's the same as:

```
int? myAge = AskMyAge();
if (myAge == null) myAge = 29;
```

Personally, I'm not a fan of the Null Coalescing Operator (??) because I feel that it is reduces the readability of the code.

And finally (for now),

The Null Conditional Operator - Elvis is in the building!

This one is kind of useful. The most common problem with nulls is when someone tries to use the dot operator on an Object when that object is set to null. That causes a Null Pointer Exception (NPE), the bane of every junior programmers existence.

If you are lazy, you can use "?." instead of "." to deference an option that may or may not be null. If it is null, you will not get an NPE. You can also combine the Null Conditional Operator with the Null Coalescing Operator to get a joyous symphony of null-i-ness:

```
Console.WriteLine($"That string is {input?.length ?? 0} characters long.");
```

as opposed to this:

```
int length = (input == null) ? 0 : input.length;
Console.WriteLine($"That string is {length} characters long.");
```

Chip says: *Given my passion for code that is clear, easy to read and easy to understand, I prefer to see the test for null in actual the code. It also makes setting a breakpoint much easier.*

Side Note: This is sometimes called the "Elvis operator" because of its similarity to the emoticon for Elvis (?:)