CHAPTER 5

Strings

The *string* data type is used to store string constants. They are delimited by double quotes.

```
string a = "Hello";
```

String Concatenation

The concatenation operator (+) can combine strings together. It also has an accompanying assignment operator (+=), which appends a string to another and creates a new string.

```
string b = a + " World"; // Hello World
a += " World";           // Hello World
```

When one of the operands is not of a string type, the concatenation operator will implicitly convert the non-string type into a string, making the following assignment valid.

```
int i = 1;
string c = i + " is " + 1; // 1 is 1
```

The string conversion is performed implicitly using the `ToString` method. All types in .NET have this method, which provides a string representation of a variable or expression. As seen in the next example, the string conversion can also be made explicitly.

```
string d = i.ToString() + " is " + 1.ToString(); // 1 is 1
```


M. Olsson, *C# 8 Quick Syntax Reference*, https://doi.org/10.1007/978-1-4842-5577-3_5

Another way to compile strings is to use string interpolation. This feature was added in C# 6.0 and enables expressions placed inside curly brackets to be evaluated within a string. To perform string interpolation, a dollar sign ($) is placed before the string.

```
string s1 = "Hello";
string s2 = "World";
string s = $"{s1} {s2}"; // Hello World
```

Escape Characters

A statement can be broken up across multiple lines, but a string constant must be on a single line. In order to divide it, the string constant has to first be split up using the concatenation operator.

```
string myString
         = "Hello " +
           "World";
```

To add new lines into the string itself, the escape character (\n) is used.

```
string myString = "Hello\nWorld";
```

This backslash notation is used to write special characters, such as a backslash or double quote. Among the special characters is also a Unicode character notation for writing any character.

Character	Meaning	Character	Meaning
\n	Newline	\f	Form feed
\t	Horizontal tab	\a	Alert sound
\v	Vertical tab	\'	Single quote
\b	Backspace	\"	Double quote
\r	Carriage return	\\	Backslash
\0	Null character	\uFFFF	Unicode character (four-digit hex number)

Escape characters can be ignored by adding an @ symbol before the string. This is called a *verbatim string* and can be used to make file paths more readable, for example.

```
string s1 = "c:\\Windows\\System32\\cmd.exe";
string s2 = @"c:\Windows\System32\cmd.exe";
```

String Compare

The way to compare two strings is simply by using the equal to operator (==). This will not compare the memory addresses, as in some other languages such as Java.

```
string greeting = "Hi";
bool b = (greeting == "Hi"); // true
```

String Members

The string type is an alias for the `String` class. As such, it provides a multitude of methods related to strings, for example, methods like `Replace`, `Insert`, and `Remove`. An important thing to note is that there are no methods for changing a string. Methods that appear to modify a string actually always return a completely new string. This is because the `String` class is immutable. The content of a string variable cannot be changed unless the whole string is replaced.

```
string a = "String";
string b = a.Replace("i", "o"); // Strong
       b = a.Insert(0, "My ");  // My String
       b = a.Remove(0, 3);      // ing
       b = a.Substring(0, 3);   // Str
       b = a.ToUpper();         // STRING
int    i = a.Length;            // 6
```

StringBuilder Class

`StringBuilder` is a mutable string class. Because of the performance cost associated with replacing a string, the `StringBuilder` class is a better alternative when a string needs to be modified many times.

```
System.Text.StringBuilder sb = new
System.Text.StringBuilder("Hello");
```

The class has several methods that can be used to manipulate the actual content of a string, such as `Append`, `Remove`, and `Insert`.

```
sb.Append(" World");   // Hello World
sb.Remove(0, 5);       // World
sb.Insert(0, "Bye");   // Bye World
```

To convert a `StringBuilder` object back into a regular string, you use the `ToString` method.

```
string s = sb.ToString(); // Bye World
```