Session 19

String

- Special Escaped characters

Escaped character	Purpose
\"	Include a double quote in a string literal.
\'	Include a single quote in a char literal.
\\	Insert a backslash.
\n	New line.
\r	Carriage return.
\t	Tab.

Escaped character	Purpose
`\0`	The character represented by the `char` with value zero (not the character `'0'`).
`\a`	Alert or "Bell". Back in the dim and distant past, terminals didn't really have sound, so you couldn't play a great big *.wav* file beautifully designed by Robert Fripp every time you wanted to alert the user to the fact that he had done something a bit wrong. Instead, you sent this character to the console, and it beeped at you, or even dinged a real bell (like the line-end on a manual typewriter). It still works today, and on some PCs there's still a separate speaker just for making this old-school beep. Try it, but be prepared for unexpected retro-side effects like growing enormous sideburns and developing an obsession with disco.

`\b`	Backspace. Yes, you can include backspaces in your string.
	
	Write:
	
	`"Hello world\b\b\b\b\bdolly"`
	
	to the console, and you'll see:
	
	`Hello dolly`
	
	Not all rendering engines support this character, though. You can see the same string rendered in a WPF application in Figure 10-1. Notice how the backspace characters have been ignored.
	
	Remember: output mechanisms can interpret individual characters differently, even though they're the *same* character, in the *same* string.

\f	Form feed. Another special character from yesteryear. This used to push a whole page worth of paper through the printer. This is somewhat less than useful now, though. Even the console doesn't do what you'd expect.
	
	If you write:
	
	`"Hello\fworld"`
	
	to the console, you'll see something like:
	
	`Hello♀world`
	
	Yes, that is the symbol for "female" in the middle there. That's because the original IBM PC defined a special character mapping so that it could use some of these characters to produce graphical symbols (like male, female, heart, club, diamond, and spade) that weren't part of the regular character set. These mappings are sometimes called *code pages*, and the default code page for the console (at least for U.S. English systems) incorporates those original IBM definitions. We'll talk more about code pages and encodings later.

\v	Vertical quote. This one looks like a "male" symbol (♂) in the console's IBM-emulating code page.

Formatting data for output

- Converting numbers to string with ToString
- Calling ToString on a decimal
- Standard Numeric Format Strings
  - Currency format
  - Specifying decimal places with currency format
- Decimal
  - Decimal format with explicit precision
  - Decimal format with unspecified precision

Formatting data for output

- Hexadecimal
  - Hexadecimal format with explicit precision
- Exponential form
- Fixed point
- General format
- Numeric
- Percent
- Round trip

Custom numeric format

- The # symbol, which represents an optional digit placeholder; if the digit in this position would have been a leading or trailing 0, it will be omitted.
- The 0 symbol, which represents a required digit placeholder; the string is padded with a 0 if the place is not needed.
- The . (dot) symbol, which represents the location of the decimal point.
- The , (comma) symbol, which performs two roles: it can enable digit grouping, and it can also scale the number down.

Dates and times

- DateTimeOffset
- Showing the date in various formats
- Getting just the time
- Getting both the time and date
- Round-trip DateTime format
- Universal sortable format
- Formatting the day

Converting Strings to other types

- Converting a string to an int
- Convert a nonnumeric string to a number
- Avoiding exceptions with TryParse
- Formatting using String.Format
- Culture Sensitivity
  - Showing available cultures
  - Formatting numbers for different cultures
- Exploring Formatting rules
  - Modifying the decimal separator

Accessing Characters by index

- Retrieving characters with a string's indexer
- Using strings as keys in a dictionary
- Getting a range of characters (using Substring)
- Composing Strings
- String.Concat explicitly
- String Join
- String Splitting
- Upper and Lower case
- Cleaning up input

StringBuilder

- Capacity – Length
- Appending
- Replace
- Find
- Empty
- Trimming
- Check character type
- Encoding and decoding

Files and Streams