Types in code

Type safety

C# is a **type-safe** language which means you use different types to represent different data.

Most basic types should be clear in their usage:

- Whole numbers
  - Used whenever you want to count something
- Fractional numbers
  - Used a lot in programming for all numbers which are not whole numbers
- True or False flag
  - Used whenever you have something in your game that can have two states
- Single letters
  - Not used too often, contains a numeric value of a character
- Text
  - Used to store text, e.g a players name: “Merlin”

Whole Numbers

```
int myAge = 99;

int catsInterestInPeople = -10;

int numberOfPets = 7;
```

Great for when you want to store information that might need counting.

For example:

- Players level = 77;
- Number of items in inventory = 16;
- Your age = 18;
- Motivation = -1;

Factional Numbers

```
float worldCoordinate = 52.567f;

float multiplier = 3.25f;

float interestRate = .5f;

double bacterialSize = 24e-10d;

double atomSize = 0.00000000000001d;
```

Perfect for coordinates or progression and is widely used in programming to:

- Update the players world position
- Set the progression of a quest
- Give yourself an interest rate
- Or make use of a certain multiplier for damage

A fractional number has a notation after its number in such a way:

- **12.1234f** **(a floating point number)**
- **99.89d** **(a decimal point number)**

We will get more into those later, but for now know that these numbers are inaccurate approximations for memory reasons.
Therefore you need to avoid comparing for equality and avoid very large numbers, positive and negative.

True or False flag

```
bool isPlayerDead = false;

bool lovesProgramming = true;

bool isDoorOpen = false;

bool gameOver = true;
```

Think about a game where the player is able to open and close a door, would it not be great to have a flag that can store that state given a true or false value?

With booleans you can do just that as they represent either a true or false value, which can be used to check for a condition -> Is the door closed?

It can be used for a multitude of things, like checking if the player is dead, or if you love programming :)

Single Letters

```
char c = 'C';
char a = 'A';
int result = c-a; // = 2
Console.WriteLine(result);
```

A single character container, not very often used but it has some places where it can be beneficial instead of a text.

Maybe you want to have a sign for the type of currency which is used?

As a fun thing, they are internally represented as a number and you can actually do math with them!

Try it yourself and checkout what results you get :)

Text

```
string name = "Fredrik";

Console.WriteLine("Hello, World.\nThis is a new line.");
Console.WriteLine("If you want to print \"Quotes\", escape them.");
Console.WriteLine("Better put some\t\tDistance.");

string starWarsIntro = @"Star Wars
Episode IV:
A New Hope";

string response = $"Welcome, {name}!";
```

Text or the type (string) is a great way to store texts that you might need to use for an item description or to save a players name.

There are a few characters that are special:

- “ quotes
- \ forward slashes
- and escape characters sequences **\n** or **\t**

In order to write a quote you need to escape the string with the escape character -> ****.

With it you can add a new line **\n**, or a indentation of a tab **\t.**

If you want to make a multiline string then you can use the @ symbol.

And lastly you can interpolate string and use variable within a set of curly brackets with the dollar sign to greet a user once they input their name:

- string response = $”Welcome, {name}!”;

Goal

See if you can create this output from the information you've learnt in these slides.

```
Hello World
0.3
True
```

- Create a console project called E6DataTypes
- Assign the value **Hello World** to a variable names **outputString**
- Then print it to the console
- Assign **0.3** to a **float** variable named **decimalValue**
- Print it to the console
- Assign the value **true** to a **bool** variable names **isAwesome**
- Print it to the console