C# Classes and Objects

‹ Previous Next ›

Classes and Objects

You learned from the previous chapter that C# is an object-oriented programming language.

Everything in C# is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the `class` keyword:

```
class Car
{
  string color = "red";
}
```

When a variable is declared directly in a class, it is often referred to as a **field** (or attribute).

It is not required, but it is a good practice to start with an uppercase first letter when naming classes. Also, it is common that the name of the C# file and the class matches, as it makes our code organized. However it is not required (like in Java).

Create an Object

An object is created from a class. We have already created the class named `Car`, so now we can use this to create objects.

To create an object of `Car`, specify the class name, followed by the object name, and use the keyword `new`:

Example

Create an object called "`myObj`" and use it to print the value of `color`:

```
{
  string color = "red";

  static void Main(string[] args)
  {
    Car myObj = new Car();
    Console.WriteLine(myObj.color);
  }
}
```

Run example »

Note that we use the dot syntax (`.`) to access variables/fields inside a class (`myObj.color`). You will learn more about fields in the next chapter.

Multiple Objects

You can create multiple objects of one class:

Example

Create two objects of `Car` :

```
{
  string color = "red";
  static void Main(string[] args)
  {
    Car myObj1 = new Car();
    Car myObj2 = new Car();
    Console.WriteLine(myObj1.color);
    Console.WriteLine(myObj2.color);
  }
}
```

Run example »

Using Multiple Classes

You can also create an object of a class and access it in another class. This is often used for better organization of classes (one class has all the fields and methods, while the other class holds the `Main()` method (code to be executed)).

- Car.cs
- Program.cs

Car.cs

```
class Car
{
```

Program.cs

```
class Program
{
  static void Main(string[] args)
  {
    Car myObj = new Car();
    Console.WriteLine(myObj.color);
  }
}
```

Run example »

Did you notice the `public` keyword? It is called an **access modifier**, which specifies that the `color` variable/field of `Car` is accessible for other classes as well, such as `Program`.

You will learn much more about **access modifiers** and **classes/objects** in the next chapters.

‹ Previous Next ›

COLOR PICKER

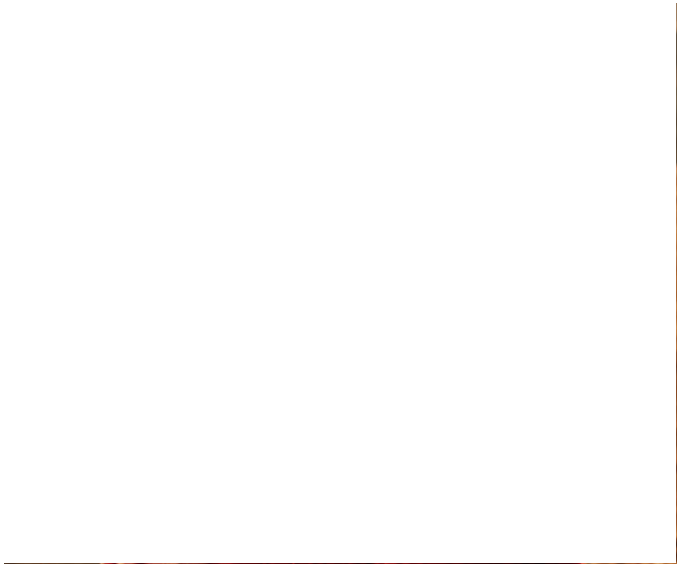
HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes
Google Maps
Range Sliders
Tooltips
Slideshow
Filter List
Sort List

SHARE

HTML
CSS
JavaScript
SQL
Python
PHP
jQuery
Bootstrap
XML

Read More »



JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
jQuery Tutorial
Java Tutorial
C++ Tutorial

JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
jQuery Reference
Java Reference
Angular Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
jQuery Examples
Java Examples
XML Examples

Web Certificates

HTML Certificate
CSS Certificate
JavaScript Certificate
SQL Certificate
Python Certificate
jQuery Certificate
PHP Certificate
Bootstrap Certificate
XML Certificate

Get Certified »