



HTML

CSS

JAVASCRIPT

MORE ▼



# C# Constructors

[< Previous](#)[Next >](#)

## Constructors

A constructor is a **special method** that is used to initialize objects. The advantage of a constructor, is that it is called when an object of a class is created. It can be used to set initial values for fields:

### Example

Create a constructor:

```
// Create a Car class
class Car
{
```

```
public string model; // Create a field

// Create a class constructor for the Car class
public Car()
{
    model = "Mustang"; // Set the initial value for model
}

static void Main(string[] args)
{
    Car Ford = new Car(); // Create an object of the Car Class (this will call the constructor)
    Console.WriteLine(Ford.model); // Print the value of model
}

// Outputs "Mustang"
```

[Run example »](#)

Note that the constructor name must **match the class name**, and it cannot have a **return type** (like `void` or `int`).

Also note that the constructor is called when the object is created.

All classes have constructors by default: if you do not create a class constructor yourself, C# creates one for you. However, then you are not able to set initial values for fields.

**Constructors save time!** Take a look at the last example on this page to really understand why.

# Constructor Parameters

Constructors can also take parameters, which is used to initialize fields.

The following example adds a `string modelName` parameter to the constructor. Inside the constructor we set `model` to `modelName` (`model=modelName`). When we call the constructor, we pass a parameter to the constructor (`"Mustang"`), which will set the value of `model` to `"Mustang"`:

## Example

```
class Car
{
    public string model;

    // Create a class constructor with a parameter
    public Car(string modelName)
    {
        model = modelName;
    }

    static void Main(string[] args)
    {
        Car Ford = new Car("Mustang");
        Console.WriteLine(Ford.model);
    }
}

// Outputs "Mustang"
```

[Run example »](#)

You can have as many parameters as you want:

## Example

```
class Car
{
    public string model;
    public string color;
    public int year;

    // Create a class constructor with multiple parameters
    public Car(string modelName, string modelColor, int modelYear)
    {
        model = modelName;
        color = modelColor;
        year = modelYear;
    }

    static void Main(string[] args)
    {
        Car Ford = new Car("Mustang", "Red", 1969);
        Console.WriteLine(Ford.color + " " + Ford.year + " " + Ford.model);
    }
}

// Outputs Red 1969 Mustang
```

[Run example »](#)

**Tip:** Just like other methods, constructors can be **overloaded** by using different numbers of parameters.

## Constructors Save Time

When you consider the example from the previous chapter, you will notice that constructors are very useful, as they help reducing the amount of code:

Without constructor:

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Car Ford = new Car();
        Ford.model = "Mustang";
        Ford.color = "red";
        Ford.year = 1969;

        Car Opel = new Car();
        Opel.model = "Astra";
    }
}
```

With constructor:

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Car Ford = new Car("Mustang", "Red", 1969);
        Car Opel = new Car("Astra", "White", 2005);

        Console.WriteLine(Ford.model);
        Console.WriteLine(Opel.model);
    }
}
```

```
Opel.color = "white";  
Opel.year = 2005;
```

[Run example »](#)[< Previous](#)[Run example »](#)[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



## CERTIFICATES

HTML

CSS

JavaScript

SQL

Python

PHP

jQuery

Bootstrap

XML

[Read More »](#)





[REPORT ERROR](#)[PRINT PAGE](#)[FORUM](#)[ABOUT](#)

## Top Tutorials

- [HTML Tutorial](#)
- [CSS Tutorial](#)
- [JavaScript Tutorial](#)
- [How To Tutorial](#)
- [SQL Tutorial](#)
- [Python Tutorial](#)
- [W3.CSS Tutorial](#)
- [Bootstrap Tutorial](#)
- [PHP Tutorial](#)
- [jQuery Tutorial](#)
- [Java Tutorial](#)
- [C++ Tutorial](#)

## Top Examples

- [HTML Examples](#)
- [CSS Examples](#)
- [JavaScript Examples](#)
- [How To Examples](#)

## Top References

- [HTML Reference](#)
- [CSS Reference](#)
- [JavaScript Reference](#)
- [SQL Reference](#)
- [Python Reference](#)
- [W3.CSS Reference](#)
- [Bootstrap Reference](#)
- [PHP Reference](#)
- [HTML Colors](#)
- [jQuery Reference](#)
- [Java Reference](#)
- [Angular Reference](#)

## Web Certificates

- [HTML Certificate](#)
- [CSS Certificate](#)
- [JavaScript Certificate](#)
- [SQL Certificate](#)

[SQL Examples](#)  
[Python Examples](#)  
[W3.CSS Examples](#)  
[Bootstrap Examples](#)  
[PHP Examples](#)  
[jQuery Examples](#)  
[Java Examples](#)  
[XML Examples](#)

[Python Certificate](#)  
[jQuery Certificate](#)  
[PHP Certificate](#)  
[Bootstrap Certificate](#)  
[XML Certificate](#)

[Get Certified »](#)

---

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2020 by Refsnes Data. All Rights Reserved.

Powered by W3.CSS.

