

## note

### Single line *if* statements

As well as using curly brackets `{}`, you can put an *if* statement on a single line. For example:

```
if (Page.IsValid)
    Response.Write("Validation OK");
```

Of course, this is useful only if you have just one line of code you intend your *if* statement to run.

Some programmers prefer to use the curly brackets even when they want to run only a single line of code, as this is more consistent and makes the code easier to understand.

## important

### The = assignment operator and the == equality operator

Some programming languages use the equals sign (=) for both assignment and equality.

C# uses the double equals sign (==) for equality and the single equals sign (=) for assignment.

#### Examples

```
AcceptedTerms = true
```

Assigns the value of *true* to the *AcceptedTerms* Boolean variable.

```
AcceptedTerms == true
```

Checks whether the value of the *AcceptedTerms* variable is equal to *true*. Returns *true* if it does, and *false* if it doesn't.

If you are not completing the course incrementally use the sample file: **Lesson 7-1** to begin this lesson.

Sample files with the starting point for each lesson are also provided for all of the other lessons in this session.

# Lesson 7-1: Use the *if* statement

- 1 Open *ShiningStone* from your sample files folder.
- 2 Open the code-behind file of *buy.aspx*.
- 3 Create an *if* statement.
  1. Remove all existing code from the *ButtonSubmitOrder\_Click* event handler.
  2. Add the following code to the event handler:

```
bool AcceptedTerms = CheckBoxAcceptTerms.Checked;
```

```
protected void ButtonSubmitOrder_Click(object sender, EventArgs e)
{
    bool AcceptedTerms = CheckBoxAcceptTerms.Checked;
}
```

This code should make sense to you by now. It creates a Boolean variable called *AcceptedTerms* which will be *true* if the *CheckBoxAcceptTerms* control is checked and *false* if it is not.

You created the *CheckBoxAcceptTerms* control in the session 4 exercise.

3. Add the following code on the next line:

```
if (AcceptedTerms == true)
{
}
```

This is the first time you've seen the == equality operator. See sidebar for more on this.

This is a very simple *if* statement. As you can see, the code is very similar to the code you used to create methods. Everything inside the curly brackets `{ }` will only run if the *AcceptedTerms* variable has a value of *true*.

In other words, the code inside the curly brackets `{ }` will only run if the user has checked the *CheckBoxAcceptTerms* control.


4. Add the following code inside the *if* statement (between `{ }`):

```
Response.Write("Terms OK");
```

```
protected void ButtonSubmitOrder_Click(object sender, EventArgs e)
{
    bool AcceptedTerms = CheckBoxAcceptTerms.Checked;
    if (AcceptedTerms == true)
    {
        Response.Write("Terms OK");
    }
}
```

5. View *buy.aspx* in your browser.
6. Complete the form (without checking *I accept the terms and conditions*) and then click the *Submit Order* button.

The page will post back, but nothing interesting will happen.

 Shining Stone Jewelers

Terms OK

## important

### Shorter *if* statements

The first *if* statement you create in this lesson is a rather long-winded way of checking if the *CheckBoxAcceptTerms* control is checked, but it's the easiest to understand.

You could have shortened the code to:

```
if (AcceptedTerms)
{
}
```


An *if* statement always looks for a *true* result by default, so this will do exactly the same thing as `== true`.

You could shorten the *if* statement even further to:

```
if
(CheckBoxAcceptTerms.Checked)
{
}
```

This works because the *Checked* property is itself a *bool* value.

*if* statements always work with *bool* values. They run if the *bool* is *true* and don't if the *bool* is *false*.

 Shining Stone Jewelers

Terms OKValidation OK

7. Check the *I accept the terms and conditions* box.
8. Click *Submit Order* again.

This time the logical test in the *if* statement returns true and *Terms OK* appears on the page.

## 4 Create a nested *if* statement.

As you can see, it's quite simple to test for a single condition. You can make your program's logic more complex by putting *if* statements inside other *if* statements.

Just like putting HTML tags inside each other, this is known as *nesting*.

1. Close your browser and return to the code-behind file of *buy.aspx*.
2. Add the following code inside the *if* statement you created before:

```
if (Page.IsValid)
{
    Response.Write("Validation OK");
}
```

```
protected void ButtonSubmitOrder_Click(object sender, EventArgs e)
{
    bool AcceptedTerms = CheckBoxAcceptTerms.Checked;
    if (AcceptedTerms == true)
    {
        Response.Write("Terms OK");
        if (Page.IsValid)
        {
            Response.Write("Validation OK");
        }
    }
}
```

As you may remember from *Lesson 4-8: Use the RequiredFieldValidator control*, *Page.IsValid* is the server-side confirmation that your form's validation controls are happy with the form's content.

The *if* statement will now check if the *AcceptedTerms* variable is *true* and will only check the *Page.IsValid* property if so.

You might want to do this if you intend to reject the user's order if they do not accept the terms and conditions (making any further validation checks pointless).

3. View *buy.aspx* in your browser.
4. Complete the form without checking the *CheckBoxAcceptTerms* control and click the *Submit Order* button.

Nothing happens. The *CheckBoxAcceptTerms* control wasn't checked so neither of the *if* statements succeed.

5. Check the *CheckBoxAcceptTerms* control and submit the form again.

This time both *if* statements succeed and the results are shown on the page.

6. Close your web browser.