1overeenkomst, verdrag

2het geheel van als passend aanvaarde vormen, gebruiken

3samenkomst, vergadering

Code conventie:

Code afspraken (comments indents etc.)

Waarom zou je die gebruiken

Om het overzichtelijk te houden voor elke programmeur

Conventies

*// No:*

$a=$b+$c;

*// Yes:*

$a = $b + $c;

**if** ( $foo == $bar ) {

**echo** "<div>Hello world</div>";

}

*// This:*

**if** ( $foo == 'bar' ) {

**echo** 'Hello world';

} **else** **if** ( $foo == 'Bar' ) {

**echo** 'Hello world';

} **else** **if** ( $baz == $foo ) {

**echo** 'Hello baz';

} **else** {

**echo** 'Eh?';

}

*// Is actually equivalent to:*

**if** ( $foo == 'bar' ) {

**echo** 'Hello world';

} **else** {

**if** ( $foo == 'Bar' ) {

**echo** 'Hello world';

} **else** {

**if** ( $baz == $foo ) {

**echo** 'Hello baz';

} **else** {

**echo** 'Eh?';

}

}

}

Overview

Files MUST use only <?php and <?= tags.

Files MUST use only UTF-8 without BOM for PHP code.

Files SHOULD either declare symbols (classes, functions, constants, etc.) or cause side-effects (e.g. generate output, change .ini settings, etc.) but SHOULD NOT do both.

Namespaces and classes MUST follow an “autoloading” PSR: [PSR-0, PSR-4].

Class names MUST be declared in StudlyCaps.

Class constants MUST be declared in all upper case with underscore separators.

Method names MUST be declared in camelCase.

Files

2.1. PHP Tags

PHP code MUST use the long <?php ?> tags or the short-echo <?= ?> tags; it MUST NOT use the other tag variations.

2.2. Character Encoding

PHP code MUST use only UTF-8 without BOM.

2.3. Side Effects

A file SHOULD declare new symbols (classes, functions, constants, etc.) and cause no other side effects, or it SHOULD execute logic with side effects, but SHOULD NOT do both.

The phrase “side effects” means execution of logic not directly related to declaring classes, functions, constants, etc., merely from including the file.

“Side effects” include but are not limited to: generating output, explicit use of require or include, connecting to external services, modifying ini settings, emitting errors or exceptions, modifying global or static variables, reading from or writing to a file, and so on.

The following is an example of a file with both declarations and side effects; i.e, an example of what to avoid:

<?php

// side effect: change ini settings

ini\_set('error\_reporting', E\_ALL);

// side effect: loads a file

include "file.php";

// side effect: generates output

echo "<html>\n";

// declaration

function foo()

{

// function body

}

The following example is of a file that contains declarations without side effects; i.e., an example of what to emulate:

<?php

// declaration

function foo()

{

// function body

}

// conditional declaration is not a side effect

if (! function\_exists('bar')) {

function bar()

{

// function body

}

}

Namespace and Class Names

Namespaces and classes MUST follow an “autoloading” PSR: [PSR-0, PSR-4].

This means each class is in a file by itself, and is in a namespace of at least one level: a top-level vendor name.

Class names MUST be declared in StudlyCaps.

Code written for PHP 5.3 and after MUST use formal namespaces.

For example:

<?php

// PHP 5.3 and later:

namespace Vendor\Model;

class Foo

{

}

Code written for 5.2.x and before SHOULD use the pseudo-namespacing convention of Vendor\_ prefixes on class names.

<?php

// PHP 5.2.x and earlier:

class Vendor\_Model\_Foo

{

}

Class Constants, Properties, and Methods

The term “class” refers to all classes, interfaces, and traits.

4.1. Constants

Class constants MUST be declared in all upper case with underscore separators. For example:

<?php

namespace Vendor\Model;

class Foo

{

const VERSION = '1.0';

const DATE\_APPROVED = '2012-06-01';

}

4.2. Properties

This guide intentionally avoids any recommendation regarding the use of $StudlyCaps, $camelCase, or $under\_score property names.

Whatever naming convention is used SHOULD be applied consistently within a reasonable scope. That scope may be vendor-level, package-level, class-level, or method-level.

4.3. Methods

Method names MUST be declared in camelCase().

* In short examples that do not include [using directives](https://msdn.microsoft.com/en-us/library/sf0df423.aspx), use namespace qualifications. If you know that a namespace is imported by default in a project, you do not have to fully qualify the names from that namespace. Qualified names can be broken after a dot (.) if they are too long for a single line, as shown in the following example.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_739c4013-2a56-4e07-8ee1-b88538cb43ca');" \o "Copy to clipboard.)

var currentPerformanceCounterCategory = new System.Diagnostics.

PerformanceCounterCategory();

* You do not have to change the names of objects that were created by using the Visual Studio designer tools to make them fit other guidelines.

**Layout Conventions**

Good layout uses formatting to emphasize the structure of your code and to make the code easier to read. Microsoft examples and samples conform to the following conventions:

* Use the default Code Editor settings (smart indenting, four-character indents, tabs saved as spaces). For more information, see [Options, Text Editor, C#, Formatting](https://msdn.microsoft.com/en-us/library/03864tbz.aspx).
* Write only one statement per line.
* Write only one declaration per line.
* If continuation lines are not indented automatically, indent them one tab stop (four spaces).
* Add at least one blank line between method definitions and property definitions.
* Use parentheses to make clauses in an expression apparent, as shown in the following code.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_338ff031-0740-4da3-8e4e-18b604c185d5');)

if ((val1 > val2) && (val1 > val3))

{

// Take appropriate action.

}

**Commenting Conventions**

* Place the comment on a separate line, not at the end of a line of code.
* Begin comment text with an uppercase letter.
* End comment text with a period.
* Insert one space between the comment delimiter (//) and the comment text, as shown in the following example.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0b0bd1f5-fed4-45f5-9c05-bc3e4ee88270');)

// The following declaration creates a query. It does not run

// the query.

* Do not create formatted blocks of asterisks around comments.

**Language Guidelines**

The following sections describe practices that the C# team follows to prepare code examples and samples.

**String Data Type**

* Use the + operator to concatenate short strings, as shown in the following code.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_87ca3137-fff2-4d8e-8a12-ec81ad348269');)

string displayName = nameList[n].LastName + ", " + nameList[n].FirstName;

* To append strings in loops, especially when you are working with large amounts of text, use a [StringBuilder](https://msdn.microsoft.com/en-us/library/system.text.stringbuilder.aspx) object.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f44cb205-285e-4eb8-8c77-7cf7c3fbfa75');)

var phrase = "lalalalalalalalalalalalalalalalalalalalalalalalalalalalalala";

var manyPhrases = new StringBuilder();

for (var i = 0; i < 10000; i++)

{

manyPhrases.Append(phrase);

}

//Console.WriteLine("tra" + manyPhrases);

**Implicitly Typed Local Variables**

* Use [implicit typing](https://msdn.microsoft.com/en-us/library/bb384061.aspx) for local variables when the type of the variable is obvious from the right side of the assignment, or when the precise type is not important.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f332fd47-08c1-4e84-9e52-1e2231c30e8e');)

// When the type of a variable is clear from the context, use var

// in the declaration.

var var1 = "This is clearly a string.";

var var2 = 27;

var var3 = Convert.ToInt32(Console.ReadLine());

* Do not use [var](https://msdn.microsoft.com/en-us/library/bb383973.aspx) when the type is not apparent from the right side of the assignment.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_6c87db46-6088-4b38-bce1-00c412621c70');)

// When the type of a variable is not clear from the context, use an

// explicit type.

int var4 = ExampleClass.ResultSoFar();

* Do not rely on the variable name to specify the type of the variable. It might not be correct.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3c60fc46-661f-46cf-ab22-43f5e13b32a5');)

// Naming the following variable inputInt is misleading.

// It is a string.

var inputInt = Console.ReadLine();

Console.WriteLine(inputInt);

* Avoid the use of var in place of [dynamic](https://msdn.microsoft.com/en-us/library/dd264741.aspx).
* Use implicit typing to determine the type of the loop variable in [for](https://msdn.microsoft.com/en-us/library/ch45axte.aspx) and [foreach](https://msdn.microsoft.com/en-us/library/ttw7t8t6.aspx) loops.

The following example uses implicit typing in a for statement.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5f362960-9fad-41f7-9b19-f28c20aa5947');)

var syllable = "ha";

var laugh = "";

for (var i = 0; i < 10; i++)

{

laugh += syllable;

Console.WriteLine(laugh);

}

The following example uses implicit typing in a foreach statement.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_4aab672e-6db0-4de1-b96f-9fa12dfc11ee');)

foreach (var ch in laugh)

{

if (ch == 'h')

Console.Write("H");

else

Console.Write(ch);

}

Console.WriteLine();

**Unsigned Data Type**

* In general, use int rather than unsigned types. The use of int is common throughout C#, and it is easier to interact with other libraries when you use int.

**Arrays**

* Use the concise syntax when you initialize arrays on the declaration line.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_08255aee-a2ab-4b29-a584-5acabbb7bba3');)

// Preferred syntax. Note that you cannot use var here instead of string[].

string[] vowels1 = { "a", "e", "i", "o", "u" };

// If you use explicit instantiation, you can use var.

var vowels2 = new string[] { "a", "e", "i", "o", "u" };

// If you specify an array size, you must initialize the elements one at a time.

var vowels3 = new string[5];

vowels3[0] = "a";

vowels3[1] = "e";

// And so on.

HTML

YES

<!DOCTYPE html>

NO

<!doctype html>

NO

<SECTION>   
  <p>This is a paragraph.</p>  
</SECTION>

YES

<section>   
  <p>This is a paragraph.</p>  
</section>

NO

<section>  
  <p>This is a paragraph.  
  <p>This is a paragraph.  
</section>

YES

<section>  
  <p>This is a paragraph.</p>  
  <p>This is a paragraph.</p>  
</section>