WIKIPEDIA

PHP syntax and semantics

The **PHP syntax and semantics** are the format (syntax) and the related meanings (semantics) of the text and symbols in the PHP programming language. They form a set of rules that define how a PHP program can be written and interpreted.

Contents

Overview

Historically, the development of PHP has been somewhat haphazard. To counter this, the PHP Framework Interop Group (FIG) has created The PHP Standards Recommendation (PSR) documents that have helped bring more standardization to the language since 2009.[1] The modern coding standards are contained in PSR-1 (http://www.php-fig.org/psr/psr-1/) (Basic Coding Standard) and PSR-2 (http://www.php-fig.org/psr/psr-2/) (Basic Style Guide).

Basic language constructs

PHP generally follows C syntax, with exceptions and enhancements for its main use in web development, which makes heavy use of string manipulation. PHP variables must be prefixed by "$". This allows PHP to perform string interpolation in double quoted strings, where backslash is supported as an escape character. No escaping or interpolation is done on strings delimited by single quotes. PHP also supports a C-like sprintf function. Code can be modularized into functions defined with keyword `function`. PHP supports an optional object oriented coding style, with classes denoted by the `class` keyword. Functions defined inside classes are sometimes called methods. Control structures include: if, while, do/while, for, foreach, and switch. Statements are terminated by a semicolon, not line endings.[2]

Delimiters

The PHP processor only parses code within its delimiters. Anything outside its delimiters is sent directly to the output and not parsed by PHP. The only open/close delimiters allowed by PSR-1[3] are "`<?php`" and "`?>`" or `<?=` and `?>`. In files containing only PHP, the closing tag should be omitted.

The purpose of the delimiting tags is to separate PHP code from non-PHP data (mainly HTML). Although rare in practice, PHP will execute code embedded in any file passed to its interpreter, including binary files such as PDF or JPEG files, or in server log files.[4][5] Everything outside the delimiters is ignored by the PHP parser and is passed through as output.[6]

These recommended delimiters create correctly formed XHTML and other XML documents.[7] This may be helpful if the source code documents ever need to be processed in other ways during the life of the software.

If proper XML validation is not an issue, and a file contains only PHP code, it is preferable to omit the PHP closing (`?>`) tag at the end of the file.[8]

Non-recommended tags

Other delimiters can be used on some servers, though most are no longer supported.[9] Examples are:

- "`<script language="php">`" and "`</script>`" (removed in PHP7)
- Short opening tags (`<?`) (configured with the `short_open_tag` ini setting)
  - A special form of the `<?` tag is `<?=`, which automatically echos the next statement. Prior to PHP 5.4.0 this was also controlled with `short_open_tag`, but is always available in later versions.
- ASP style tags (<% or <%=) (removed in PHP7)

Control structures

Conditional statements

If ... else

The syntax of a PHP If ... else statement is as follows:

```
<?php
if (condition) {
    // statements;
} elseif (condition2) {
    // statements;
} else {
    // statements;
}
?>
```

For single statements, the brackets may be omitted and the if optionally condensed to a single line:

```
<?php
if (condition) dosomething();
elseif (condition2) dosomethingelse();
else doyetathirdthing();
?>
```

Switch

An example of the syntax of a PHP switch statement is as follows:

```
<?php
switch (expr) {
    case 0:
      // statements;
      break;
    case 1:
      // statements;
      break;
    case 2:
      // statements;
      break;
    default:
      // statements;
}
?>
```

Note that unlike in C, values in case statement can be any type, not just integers.[10]

Loops

For loop

The PHP syntax of a for loop is as follows:

```
<?php
for (initialization; condition; afterthought) {
    // statements;
}
?>
```

For each loop

While loop

The syntax for a PHP while loop is as follows:

```
<?php
while (condition) {
    // statements;
}
?>
```

Do while loop

The syntax for a PHP Do while loop is as follows:

```
<?php
do {
    // statements;
} while (condition);
?>
```

Variables and comments

Variables are prefixed with a dollar symbol and a type does not need to be specified in advance. Unlike function and class names, variable names are case-sensitive. Both double-quoted (`""`) and heredoc strings allow the ability to embed a variable's value into the string.[11] As in C, variables may be cast to a specific type by prefixing the type in parentheses. PHP treats newlines as whitespace, in the manner of a free-form language. The concatenation operator is `.` (dot). Array elements are accessed and set with square brackets in both associative arrays and indexed arrays. Curly brackets can be used to access array elements, but not to assign.

PHP has three types of comment syntax: `/* */` which serves as block comments, and `//` as well as `#` which are used for inline comments.[12] Many examples use the `print` function instead of the `echo` function. Both functions are nearly identical; the major difference being that `print` is slower than `echo` because the former will return a status indicating if it was successful or not in addition to text to output, whereas the latter does not return a status and only returns the text for output.[13]

The usual "Hello World" code example for PHP is:[14]

```
<?php
echo "Hello World!\n";
?>
```

The example above outputs the following:

```
Hello World!
```

Instead of using `<?php` and the echo statement, an optional "shortcut" is the use of `<?=` instead of `<?php` which implicitly echoes data. For example, to show the page_title:

```
<!DOCTYPE html>
<html>
    <head>
        <title><?=$page_title;?></title>
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html>
```

The above example also illustrates that text not contained within enclosing PHP tags will be directly output. So the simplest form of Hello World in PHP is a plain text file containing "Hello World".

Alternative syntax for control structures

PHP offers an alternative syntax using colons rather than the standard curly-brace syntax (of "{...}"). This syntax affects the following control structures: if, while, for, foreach, and switch. The syntax varies only slightly from the curly-brace syntax. In each case the opening brace ({) is replaced with a colon (:) and the close brace is replaced with endif;, endwhile;, endfor;, endforeach;, or endswitch;, respectively.[15] Mixing syntax styles within the same control block is not supported. An example of the syntax for an if/elseif statement is as follows:

```php
<?php
if (condition):
     // code here
elseif (condition):
     // code here
else:
     // code here
endif;
?>
```

This style is sometimes called template syntax, as it is often found easier to read when combining PHP and HTML or JavaScript for conditional output:

```php
<html>
<?php if ($day == 'Thursday'): ?>
  <div>Tomorrow is Friday!</div>
<?php elseif ($day == 'Friday'): ?>
  <div>TGIF</div>
<?php else: ?>
  <div>ugh</div>
<?php endif; ?>
</html>
```

Data types

PHP stores whole numbers in a platform-dependent range. This range is typically that of 32-bit signed integers. Integer variables can be assigned using decimal (positive and negative), octal and hexadecimal notations. Real numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation.[16]

PHP has a native Boolean type, named "boolean", similar to the native Boolean types in Java and C++. Using the Boolean type conversion rules, non-zero values are interpreted as true and zero as false, as in Perl.[16]

The null data type represents a variable that has no value. The only value in the null data type is *NULL*.[16] Variables of the "resource" type represent references to resources from external sources. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension. Examples include file, image and database resources.[16]

Arrays can contain mixed elements of any type, including resources, objects. Multi-dimensional arrays are created by assigning arrays as array elements. PHP has no true array type. PHP arrays are natively sparse and associative. Indexed arrays are simply hashes using integers as keys. Objects can syntactically be used as Arrays.[16]

Functions

PHP has hundreds of base functions and thousands more from extensions. Prior to PHP version 5.3.0, functions are not first-class functions and can only be referenced by their name, whereas PHP 5.3.0 introduces closures.[17] User-defined functions can be created at any time and without being prototyped.[17] Functions can be defined inside code blocks, permitting a run-time decision as to whether or not a function should be defined. There is no concept of local functions. Function calls must use parentheses with the exception of zero argument class constructor functions called with the PHP `new` operator, where parentheses are optional.

An example function definition is the following:

```
<?php
function hello($target='World')
{
    echo "Hello $target!\n";
}

hello(); // outputs "Hello World!"
hello('Wikipedia'); // outputs "Hello Wikipedia!"
?>
```

PHP supports true anonymous functions as of version 5.3.[17] In previous versions, PHP only supported quasi-anonymous functions through the `create_function()` function.

Function calls may be made via variables, where the value of a variable contains the name of the function to call. This is illustrated in the following example:

```
<?php
function hello()
{
    return 'Hello';
}

function world()
{
    return "World!";
}

$function1 = 'hello';
$function2 = 'world';

echo "{$function1()} {$function2()}";
?>
```

A default value for parameters can be assigned in the function definition, but PHP does not support named parameters or parameter skipping.[18] Some core PHP developers have publicly expressed disappointment with this decision.[19] Others have suggested workarounds for this limitation.[20]

Objects

Basic object-oriented programming functionality was added in PHP 3.[21] Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance.[22] In previous versions of PHP, objects were handled like primitive types.[22] The drawback of this method was that the whole object was copied when a variable was assigned or passed as a parameter to a method. In the new approach, objects are referenced by handle, and not by value. PHP 5 introduced private and protected member variables and methods, along with abstract classes and final classes as well as abstract methods and final methods. It also introduced a standard way of declaring constructors and destructors, similar to that of other object-oriented languages such as C++, and a standard exception handling model. Furthermore PHP 5 added Interfaces and allows for multiple Interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. Objects implementing ArrayAccess can be used with array syntax and objects implementing Iterator or IteratorAggregate can be used with the foreach language construct. The static method and class variable features in Zend Engine 2 do not work the way some would expect. There is no virtual table feature in the engine, so static variables are bound with a name instead of a reference at compile time.[23]

This example shows how to define a class, `foo`, that inherits from class `bar`. The function `mystaticfunc` is a public static function that can be called with `foo::mystaticfunc();`.

```php
class Foo extends Bar
{
    function __construct()
    {
        $doo = "wah dee dee";
    }

    public static function myStaticMethod()
    {
        $dee = "dee dee dum";
    }
}
```

If the developer creates a copy of an object using the reserved word *clone*, the Zend engine will check if a `__clone()` method has been defined or not. If not, it will call a default `__clone()` which will copy the object's properties. If a `__clone()` method is defined, then it will be responsible for setting the necessary properties in the created object. For convenience, the engine will supply a function that imports the properties of the source object, so that the programmer can start with a by-value replica of the source object and only override properties that need to be changed.[24]

See also

- Hypertext Markup Language (HTML)
- Template engine (web)

References

1. PSR-Huh? (http://code.tutsplus.com/tutorials/psr-huh--net-29314)
2. "Instruction separation" (http://www.php.net/basic-syntax.instruction-separation). The PHP Group. Retrieved 2008-03-16.
3. http://www.php-fig.org/psr/psr-1/
4. "Code injection – a simple PHP virus carried in a JPEG image" (http://php.webtutor.pl/en/2011/05/13/php-code-injection-a-simple-virus-written-in-php-and-carried-in-a-jpeg-image/).
5. "The Hacker's Handbook: The Strategy Behind Breaking into and Defending Networks" (https://books.google.com/books?id=AO2fsAPVC34C&pg=PA638&lpg=PA638).
6. "Your first PHP-enabled page" (http://ca3.php.net/manual/en/tutorial.firstpage.php). The PHP Group. Retrieved 2008-02-25.
7. Bray, Tim; et al. (26 November 2008). "Processing Instructions" (http://www.w3.org/TR/2008/REC-xml-20081126/#sec-pi). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C. Retrieved 2009-06-18.
8. "PHP tags" (http://php.net/manual/en/language.basic-syntax.phptags.php).
9. "PHP: Basic syntax" (http://www.php.net/manual/en/language.basic-syntax.php). The PHP Group. Retrieved 2008-02-22.
10. http://php.net/manual/en/control-structures.switch.php
11. "Variables" (http://www.php.net/manual/en/language.variables.php). The PHP Group. Retrieved 2008-03-16.
12. "Comments" (http://ca3.php.net/manual/en/language.basic-syntax.comments.php). The PHP Group. Retrieved 2008-03-16.
13. "print" (http://www.php.net/print). The PHP Group. Retrieved 2008-03-16.
14. "Hello World" (http://php.codenewbie.com/articles/php/1485/Hello_World-Page_1.html). Code Newbie. Retrieved 2008-02-25.
15. "Alternative syntax for control structures" (http://php.net/manual/en/control-structures.alternative-syntax.php). The PHP Group. Retrieved 2010-04-16.
16. "Types" (http://www.php.net/manual/en/language.types.php). The PHP Group. Retrieved 2008-03-16.
17. "Functions" (http://www.php.net/manual/en/language.functions.php). The PHP Group. Retrieved 2008-03-16.
18. "PHP 6 Dropped Items" (http://wiki.php.net/todo/backlog#dropped_items). The PHP Group. Retrieved 2009-01-09.
19. "Syntax I Miss in PHP" (http://php100.wordpress.com/2009/08/21/syntax-i-miss-in-php/). Stanislav Malyshev, Zend Technologies, Ltd. Retrieved 2009-01-09.
20. "PHP Skipped and Named Parameters" (http://www.seoegghead.com/software/php-parameter-skipping-and-named-parameters.seo). SEO Egghead Inc. Retrieved 2009-01-09.
21. "History of PHP and related projects" (http://www.php.net/history). The PHP Group. Retrieved 2008-02-25.
22. "PHP 5 Object References" (http://mjtsai.com/blog/2004/07/15/php-5-object-references/). mjtsai. Retrieved 2008-03-16.
23. "Classes and Objects (PHP 5)" (http://ca3.php.net/zend-engine-2.php). The PHP Group. Retrieved 2008-03-16.
24. "Object cloning" (http://ca3.php.net/language.oop5.cloning). The PHP Group. Retrieved 2008-03-16.