

Data Types

integer, float, boolean, string, array, object, resource, NULL

Variable Declarations

`$variablename = <value>;`

`$anothervariable =& $variablename;` (Assign by Reference)

Declare Array

`$arrayname = array();`

Initialize Array

`$arrayname = array(<value1>, <value2>, <value3>);`

`$arrayname = array(<key> => <value>, <key> => <value>);` (Define Keys)

`$multiarray = array(<key> => array(<value1>, <value2>));` (Multi-dimensional)

Common Array Functions

`sort(<array>);` (Sort array assigns new keys)

`asort(<array>);` (Sort array maintain keys)

`rsort(<array>);` (Sort array in reverse, new keys)

`arsort(<array>);` (Sort array in reverse, maintain keys)

`count(<array>);` (Count elements)

`count(<array>, COUNT_RECURSIVE);` (Count multidimensional array)

`array_push(<array>, <value>);` (Push item onto end of array)

`array_pop(<array>);` (Pop item off end of array)

Comments

`// Comment text`

`/* Multi-line comment text */`

`# Comment text`

Arithmetic Operators

`+` (Addition), `-` (Subtraction), `*` (Multiplication), `/` (Division), `%` (Modulus)

Relational Operators

`==` (Equal), `===` (Equal with type comparison), `!=` (Not equal), `<>` (Not equal), `!==` (Not Equal with type comparison), `<` (Less than), `>` (Greater than), `<=` (Less than or equal to), `>=` (Greater than or equal to)

Logical Operators

`!` (logical NOT), `&&` (logical AND), `||` (logical OR), `xor` (logical XOR)

Assignment Operators

`=` (Assign), `+=` (Addition), `-=` (Subtraction), `*=` (Multiplication), `/=` (Division), `.=` (Concatenation), `%=` (Modulus), `&=` (And), `|=` (Or), `^=` (Exclusive Or), `<<=` (Left Shift), `>>=` (Right Shift)

String Concatenation

`.` (Period)

String Manipulation

`substr(<string>, <start>, [<length>]);`

`strlen(<string>);`

`trim(<string>);`

`ltrim(<string>);` // Trim left

`rtrim(<string>);` // Trim right

`strtolower(<string>);`

`strtoupper(<string>);`

`str_replace(<search>, <replace>, <string>, [<count>]);`

`strpos(<string>, <search>);`

`strcmp(<string1>, <string2>);` (Binary safe string comparison)

`strcasecmp(<string1>, <string2>);` (Binary safe case-insensitive comparison)

`explode(<delim>, <string>, [<limit>]);` (Break string into array)

`implode(<delim>, <array>);` (Join array into string separated by delim)

Cookies

`setcookie (<cookie name>, [<value>], [<expire_time_in_secs_since_epoch>];`

`$_COOKIE['cookie name'];` (Returns value of cookie)

Sessions

`session_start();` (Create session)

`$_SESSION['key_name'] = value;` (Set session variable)

`$variablename = $_SESSION['key_name'];` (Retrieve value from session variable)

`session_destroy();` (Destroy session)

Error Handling

`try {`

`<statements that may cause error>;`

`}`

`catch (<Exception Class> $exception_name)`

`{`

`<statements to execute when error is caught>;`

`}`

Super Globals

`$GLOBALS` (Access all global variables in script)

`$_SERVER` (Access web server variables)

`$_GET` (Values passed to script through URL)

`$_POST` (Values passed to script through HTTP Post)

`$_COOKIE` (Values passed by user cookie)

`$_FILES` (Values passed by HTTP Post File Uploads)

`$_ENV` (Values passed to script via the environment)

`$_REQUEST` (Values passed by URL, HTTP Post, or user Cookies)

`$_SESSION` (Values passed through user's session)

If Else

`if (<condition 1>)`

`{ <statement 1>; }`

`elseif (<condition 2>)`

`{ <statement 2>; }`

`else`

`{ <statement 3>; }`

Inline If (Ternary)

`<condition> ? true : false;`

For Loop

`for (<initialize>; <condition>; <update>)`

`{`

`<statements>;`

`}`

For Each Loop

`foreach (<array> as [<value> | <key> => <value>])`

`{`

`<statements>;`

`[break];`

`[continue];`

`}`

While Loop

`while (<condition>)`

`{`

`<statements>;`

`}`

Do-While Loop

`do`

`{`

`<statements>;`

`} while (<condition>;)`

Switch

`switch (<expression>)`

`{`

`case <literal or type>:`

`<statements>;`

`[break];`

`case <literal or type>:`

`<statements>;`

`[break];`

`default:`

`<statements>;`

`}`

Function Structure

`function <function_name>([<parameters>])`

`{`

`<statements>;`

`[return <value>;]`

`}`

Class Structure

`class <class_name> [<extends base_class>]`

`{`

`[var | <modifiers*>] <class member variables>;`

`[<modifiers*>] function <function_name>([<parameters>])`

`{`

`<statements>;`

`}`

* Modifiers `<public | private | static>` are implemented in PHP5

Declare and Use Class

`$variable = new class_name();`

`$variable->function_name();`

`class_name::function_name();` (Static call)