Type Juggling ¶

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which the variable is used. That is to say, if a string value is assigned to variable *$var*, *$var* becomes a string. If an int value is then assigned to *$var*, it becomes an int.

An example of PHP's automatic type conversion is the multiplication operator '*'. If either operand is a float, then both operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as ints, and the result will also be an int. Note that this does *not* change the types of the operands themselves; the only change is in how the operands are evaluated and what the type of the expression itself is.

```php
<?php
$foo = "1";  // $foo is string (ASCII 49)
$foo *= 2;   // $foo is now an integer (2)
$foo = $foo * 1.3;  // $foo is now a float (2.6)
$foo = 5 * "10 Little Piggies"; // $foo is integer (50)
$foo = 5 * "10 Small Pigs";     // $foo is integer (50)
?>
```

If the last two examples above seem odd, see how numeric strings convert to integers.

To force a variable to be evaluated as a certain type, see the section on Type casting. To change the type of a variable, see the settype() function.

To test any of the examples in this section, use the var_dump() function.

> **Note**:
>
> The behaviour of an automatic conversion to array is currently undefined.
>
> Also, because PHP supports indexing into strings via offsets using the same syntax as arrayindexing, the following example holds true for all PHP versions:
>
> ```php
> <?php
> $a    = 'car'; // $a is a string
> $a[0] = 'b';   // $a is still a string
> echo $a;       // bar
> ?>
> ```
>
> See the section titled String access by character for more information.

Type Casting ¶

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```php
<?php
$foo = 10;   // $foo is an integer
$bar = (boolean) $foo;   // $bar is a boolean
?>
```

The casts allowed are:

- (int), (integer) - cast to int
- (bool), (boolean) - cast to bool
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

- (unset) - cast to NULL

(binary) casting and b prefix exists for forward support. Note that the (binary) cast is essential the same as (string), but it should not be relied upon.

The (unset) cast has been deprecated as of PHP 7.2.0. Note that the (unset) cast is the same as assigning the value NULL to the variable or call. The (unset) cast is removed as of PHP 8.0.0.

Note that tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent:

```php
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

Casting literal strings and variables to binary strings:

```php
<?php
$binary = (binary) $string;
$binary = b"binary string";
?>
```

> **Note**:
>
> Instead of casting a variable to a string, it is also possible to enclose the variable in double quotes.
>
> ```php
> <?php
> $foo = 10;            // $foo is an integer
> $str = "$foo";        // $str is a string
> $fst = (string) $foo; // $fst is also a string
>
> // This prints out that "they are the same"
> if ($fst === $str) {
>     echo "they are the same";
> }
> ?>
> ```

It may not be obvious exactly what will happen when casting between certain types. For more information, see these sections:

- Converting to boolean
- Converting to integer
- Converting to float
- Converting to string
- Converting to array
- Converting to object
- Converting to resource
- Converting to NULL
- The type comparison tables