

Conceptos básicos de PHP (I)

Ricardo Pérez López

IES Doñana, curso 2020/2021

Índice general

1. Introducción a PHP	2
1.1. Página web de PHP	2
1.2. Instalación de PHP	2
1.3. Documentación y búsqueda de información	2
2. Sintaxis básica	3
2.1. Datos e instrucciones	3
2.2. Sentencias y comandos	3
2.2.1. Comando <code>echo</code>	3
2.3. Expresiones, operadores y funciones	3
3. Funcionamiento del intérprete	3
3.1. Ejecución	3
3.1.1. Por lotes	3
3.1.2. Interactiva	3
3.2. Etiquetas <code><?php</code> y <code>?></code>	3
3.3. Modo dual de operación	3
4. Variables	3
4.1. Conceptos básicos	3
4.2. Destrucción de variables	4
4.3. Operadores de asignación por valor y por referencia	4
4.4. Variables predefinidas	4
5. Tipos básicos de datos	4
5.1. Introducción	4
5.2. Lógicos (<code>bool</code>)	4
5.2.1. Operadores lógicos	4
5.3. Numéricos	5
5.3.1. Enteros (<code>int</code>)	5
5.3.2. Números en coma flotante (<code>float</code>)	5
5.3.3. Operadores	5
5.4. Cadenas (<code>string</code>)	5
5.4.1. Operadores de cadenas	5

5.4.2. Funciones de manejo de cadenas	5
5.4.3. Extensión <i>mbstring</i>	6
5.5. Nulo (<i>null</i>)	6
6. Manipulación de datos	6
6.1. Precedencia de operadores	6
6.2. Operadores de asignación compuesta	6
6.3. Comprobaciones	6
6.3.1. De tipos	6
6.3.2. De valores	6
6.4. Conversiones de tipos	7
6.4.1. Conversión explícita (<i>casting</i>) vs. implícita (automática)	7
6.4.2. Conversión a <i>bool</i>	7
6.4.3. Conversión a <i>int</i>	7
6.4.4. Conversión a <i>float</i>	7
6.4.5. Conversión de <i>string</i> a número	7
6.4.6. Conversión a <i>string</i>	7
6.4.7. Funciones de obtención de valores	7
6.4.8. Funciones de formateado numérico	8
6.5. Comparaciones	8
6.5.1. Operadores de comparación	8
6.5.2. <i>==</i> vs. <i>===</i>	8
6.5.3. Ternario (<i>?:</i>)	8
6.5.4. Fusión de <i>null</i> (<i>??</i>)	9
6.5.5. Reglas de comparación de tipos	9
7. Constantes	9
7.1. Introducción	9
7.2. <i>define()</i> y <i>const</i>	9
7.3. Constantes predefinidas	9
7.4. <i>defined()</i>	9
8. Ejercicios	9
8.1. Actividades	9

1. Introducción a PHP

1.1. Página web de PHP

<https://php.net>

1.2. Instalación de PHP

[~/ .conf/scripts/php-install.php](#)

1.3. Documentación y búsqueda de información

<https://www.php.net/manual/es>

2. Sintaxis básica

<http://php.net/manual/es/language.basic-syntax.php>

2.1. Datos e instrucciones

2.2. Sentencias y comandos

2.2.1. Comando `echo`

<http://php.net/manual/es/function.echo.php>

2.3. Expresiones, operadores y funciones

ricpelo's note: *Ejemplos:* aritmética, `cos()`, `max()`

ricpelo's note: `print()` no es una función. Cuidado.

3. Funcionamiento del intérprete

3.1. Ejecución

3.1.1. Por lotes

3.1.2. Interactiva

3.1.2.1. `php -a`

3.1.2.2. `PsySH`

<http://psysh.org>

3.2. Etiquetas `<?php y ?>`

<https://www.php.net/manual/es/language.basic-syntax.phptags.php>

3.3. Modo dual de operación

<http://php.net/manual/es/language.basic-syntax.phpmode.php>

ricpelo's note: Se llaman *modo HTML* y *modo PHP*.

4. Variables

4.1. Conceptos básicos

<http://php.net/manual/es/language.variables.basics.php>

4.2. Destrucción de variables

<http://php.net/manual/es/function.unset.php>

4.3. Operadores de asignación por valor y por referencia

<http://php.net/manual/es/language.operators.assignment.php>

ricpelo's note: En `$b =& $a;`, `$b` **NO** está apuntando a `$a` o viceversa. Ambos apuntan al mismo lugar.

4.4. Variables predefinidas

<http://php.net/manual/es/reserved.variables.php>

ricpelo's note: `$_ENV` no funciona en la instalación actual (ver `variables_order` en `php.ini`. Habría que usar `get_env()`.

5. Tipos básicos de datos

5.1. Introducción

<http://php.net/manual/es/language.types.intro.php>

5.2. Lógicos (bool)

<http://php.net/manual/es/language.types.boolean.php>

ricpelo's note: Se escriben en minúscula: `false` y `true`.

ricpelo's note: `boolean` es sinónimo de `bool`, pero debería usarse `bool`.

5.2.1. Operadores lógicos

<http://php.net/manual/es/language.operators.logical.php>

ricpelo's note: *Cuidado:*

`false and (true && print('hola'))` no imprime nada y devuelve `false`, por lo que **el código va en cortocircuito y se evalúa de izquierda a derecha** incluso aunque el `&&` y los paréntesis tengan más prioridad que el `and`.

Otra forma de verlo es comprobar que `print('uno') and (1 + print('dos'))` escribe `unodos` (y devuelve `true`), por lo que la evaluación de los operandos del `and` se hace de izquierda a derecha aunque el `+` tenga más prioridad (y encima vaya entre paréntesis).

En el manual de PHP se dice que: “La precedencia y asociatividad de los operadores solamente determinan cómo se agrupan las expresiones, no especifican un orden de evaluación. PHP no especifica (en general) el orden en que se evalúa una expresión y se debería evitar el código que se asume un orden específico de evaluación, ya que el comportamiento puede cambiar entre versiones de PHP o dependiendo de código circundante.”

Pregunta que hice al respecto en StackOverflow.

5.3. Numéricos

5.3.1. Enteros (int)

<http://php.net/manual/es/language.types.integer.php>

ricpelo's note: `integer` es sinónimo de `int`, pero debería usarse `int`.

5.3.2. Números en coma flotante (float)

<http://php.net/manual/es/language.types.float.php>

ricpelo's note: `double` es sinónimo de `float`, pero debería usarse `float`.

5.3.3. Operadores

5.3.3.1. Operadores aritméticos

<http://php.net/manual/es/language.operators.arithmetic.php>

5.3.3.2. Operadores de incremento/decremento

<http://php.net/manual/es/language.operators.increment.php>

5.4. Cadenas (string)

<http://php.net/manual/es/language.types.string.php>

ricpelo's note: Se usa `{$var}` y no `${var}`

5.4.1. Operadores de cadenas

<http://php.net/manual/es/language.operators.string.php>

5.4.1.1. Concatenación

5.4.1.2. Acceso y modificación por caracteres

<http://php.net/manual/es/language.types.string.php#language.types.string.substr>

ricpelo's note: - `echo $a[3]`

- `$a[3] = 'x';`

5.4.1.3. Operador de incremento

<http://php.net/manual/es/language.operators.increment.php>

5.4.2. Funciones de manejo de cadenas

<http://php.net/ref.strings>

5.4.3. Extensión *mbstring*

<http://php.net/manual/en/book.mbstring.php>

ricpelo's note: - `$a[3]` equivale a `mb_substr($a, 3, 1)`

`$a[3] = 'x'`; no tiene equivalencia directa. Se podría hacer:
`$a = mb_substr($a, 2, 1) . 'x' . mb_substr($a, 4);`

5.5. Nulo (`null`)

<http://php.net/manual/es/language.types.null.php>

ricpelo's note: `is_null()` vs. `=== null`

ricpelo's note: El tipo `null` y el valor `null` se escriben en minúscula.

6. Manipulación de datos

6.1. Precedencia de operadores

<http://php.net/manual/es/language.operators.precedence.php>

6.2. Operadores de asignación compuesta

ricpelo's note: `$x <op>= $y`

6.3. Comprobaciones

6.3.1. De tipos

6.3.1.1. `gettype()`

<http://php.net/manual/en/function.gettype.php>

6.3.1.2. `is_*()`

<http://php.net/manual/es/ref.var.php>

ricpelo's note: Poco útiles en formularios, ya que sólo se reciben `strings`.

6.3.2. De valores

6.3.2.1. `is_numeric()`

<http://php.net/manual/es/function.is-numeric.php>

6.3.2.2. `ctype_*()`

<http://php.net/manual/es/book ctype.php>

6.4. Conversiones de tipos

<http://php.net/manual/es/language.types.type-juggling.php>

6.4.1. Conversión explícita (*casting*) vs. implícita (automática)

<http://php.net/manual/es/language.types.type-juggling.php#language.types.typecasting>

ricpelo's note: Conversión de cadena a número

6.4.2. Conversión a `bool`

<http://php.net/manual/es/language.types.boolean.php#language.types.boolean.casting>

6.4.3. Conversión a `int`

<http://php.net/manual/es/language.types.integer.php#language.types.integer.casting>

6.4.4. Conversión a `float`

<http://php.net/manual/es/language.types.float.php#language.types.float.casting>

6.4.5. Conversión de `string` a número

<http://php.net/manual/es/language.types.string.php#language.types.string.conversion>

ricpelo's note: ¡Cuidado!:

La documentación dice que `$x = 1 + "pepe"` o `$x = 1 + "10 pepe"` funciona, pero dependiendo del valor de `error_reporting` en `php.ini`, puede dar un **PHP Warning: A non-numeric value encountered** o un **PHP Warning: A non well formed numeric value encountered**, respectivamente.

Si `error_reporting = E_ALL`, dará el mensaje de advertencia.

Además, en PsySH no funcionará, es decir, que `$x` no se asignará al valor. En `php -a` sí funcionará (aunque da el mismo mensaje de advertencia).

Si `error_reporting = E_ALL & ~E_NOTICE`, no lo dará.

Además, funcionará tanto en PsySH como en `php -a`.

6.4.6. Conversión a `string`

<http://php.net/manual/es/language.types.string.php#language.types.string.casting>

6.4.7. Funciones de obtención de valores

ricpelo's note: Hacen más o menos lo mismo que los *casting* pero con funciones en lugar de con operadores. Puede ser interesante porque las funciones se pueden guardar, usar con *map*, *reduce*, etc.

6.4.7.1. intval()

<http://php.net/manual/es/function.intval.php>

6.4.7.2. floatval()

<http://php.net/manual/es/function.floatval.php>

6.4.7.3. strval()

<http://php.net/manual/es/function.strval.php>

6.4.7.4. boolval()

<http://php.net/manual/es/function.boolval.php>

6.4.8. Funciones de formateado numérico

6.4.8.1. number_format()

<http://php.net/manual/es/function.number-format.php>

6.4.8.2. money_format()

<http://php.net/manual/es/function.money-format.php>

`setlocale()`

ricpelo's note: `setlocale(LC_ALL, 'es_ES.UTF-8');` // Hay que poner el `*locale*` completo, con la codificación y todo (`.UTF-8`)

6.5. Comparaciones

6.5.1. Operadores de comparación

<http://php.net/manual/es/language.operators.comparison.php>

ricpelo's note: `"250" < "27"` devuelve `false`

ricpelo's note: Si se compara un número con un string o la comparación implica strings numéricos, entonces cada string es convertido en un número y la comparación realizada numéricamente.

6.5.2. == vs. ===

6.5.3. Ternario (?:)

<http://php.net/manual/es/language.operators.comparison.php#language.operators.comparison.ternary>

6.5.4. Fusión de null (??)

https://wiki.php.net/rfc/isset_ternary

ricpelo's note: Equivalente al `COALESCE()` de SQL.

6.5.5. Reglas de comparación de tipos

<http://php.net/manual/es/types.comparisons.php>

7. Constantes

7.1. Introducción

<http://php.net/manual/es/language.constants.syntax.php>

ricpelo's note: Diferencias entre constantes y variables:

Las constantes no llevan el signo dólar (\$) como prefijo.

Antes de PHP 5.3, las constantes solo podían ser definidas usando la función `define()` y no por simple asignación.

Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.

Las constantes no pueden ser redefinidas o eliminadas una vez se han definido.

Las constantes podrían evaluarse como valores escalares. A partir de PHP 5.6 es posible definir una constante de array con la palabra reservada `const`, y, a partir de PHP 7, las constantes de array también se pueden definir con `define()`. Se pueden utilizar arrays en expresiones escalares constantes (por ejemplo, `const FOO = array(1,2,3)[0];`), aunque el resultado final debe ser un valor de un tipo permitido.

7.2. `define()` y `const`

7.3. Constantes predefinidas

<http://php.net/manual/es/language.constants.predefined.php>

7.4. `defined()`

<http://php.net/manual/es/function.defined.php>

8. Ejercicios

8.1. Actividades

1. Busca información sobre la función `time()` usando, al menos, tres formas distintas.
2. Explica, con tus propias palabras, la diferencia entre:

- 2.1. Un dato y una instrucción.
 - 2.2. Una expresión y una sentencia.
 - 2.3. Una sentencia y un comando.
 - 2.4. Una función y un operador.
3. ¿Es `echo` una función? ¿A dónde acudes para saberlo?
 4. ¿Es lo mismo *modo de ejecución* que *modo de operación*? Explica cuáles son y en qué consisten los diferentes modos de ejecución y de operación en PHP.
 5. ¿Qué ventajas e inconvenientes tiene usar PsySH frente al intérprete integrado?
 6. ¿Qué tipos de asignación de variables existen en PHP? Explica sus diferencias y pon ejemplos de uso.
 7. ¿Qué son las variables predefinidas? Enumera las más importantes.
 8. Calcula el valor de las siguientes expresiones y razona por qué tienen ese valor:
 - 8.1. `false and true or 1`
 - 8.2. `1 == 1.0`
 - 8.3. `1 == 0.9999999999999999`
 - 8.4. `floor((0.1 + 0.7) * 10)`
 - 8.5. `'1' == 1`
 - 8.6. `empty('0')`
 9. ¿`$a[3]` equivale a `mb_substr($a, 3, 1)`? Razona la respuesta.
 10. Define con tus propias palabras el significado de *asociatividad* y de *prioridad*. ¿Por qué la expresión `1 == 1 == 1` es incorrecta pero `1 <= 1 == 1` es correcta (y cuál es su valor, por cierto)?

Respuestas a las preguntas

Respuestas a las preguntas

Bibliografía