

Conceptos básicos de PHP (I)

Ricardo Pérez López

IES Doñana, curso 2020/2021



1. Introducción a PHP
2. Sintaxis básica
3. Funcionamiento del intérprete
4. Variables
5. Tipos básicos de datos
6. Manipulación de datos
7. Constantes
8. Ejercicios

1. Introducción a PHP

1.1 Página web de PHP

1.2 Instalación de PHP

1.3 Documentación y búsqueda de información

1.1. Página web de PHP

1.1. Página web de PHP

`https://php.net`

1.2. Instalación de PHP

1.2. Instalación de PHP

```
~/.conf/scripts/php-install.php
```

1.3. Documentación y búsqueda de información

1.3. Documentación y búsqueda de información

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

2. Sintaxis básica

2.1 Datos e instrucciones

2.2 Sentencias y comandos

2.3 Expresiones, operadores y funciones

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`

Comando echo

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

2.3. Expresiones, operadores y funciones

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

3.2 Etiquetas `<?php y ?>`

3.3 Modo dual de operación

3.1. Ejecución

3.1.1 Por lotes

3.1.2 Interactiva

PsySH

<http://psysh.org>

3.2. Etiquetas <?php y ?>

3.2. Etiquetas <?php y ?>

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

3.3. Modo dual de operación

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

4.2 Destrucción de variables

4.3 Operadores de asignación por valor y por referencia

4.4 Variables predefinidas

4.1. Conceptos básicos

4.1. Conceptos básicos

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

4.2. Destrucción de variables

4.2. Destrucción de variables

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

4.3. Operadores de asignación por valor y por referencia

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

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

5.2 Lógicos (`bool`)

5.3 Numéricos

5.4 Cadenas (`string`)

5.5 Nulo (`null`)

5.1. Introducción

5.1. Introducción

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

5.2. Lógicos (bool)

5.2.1 Operadores lógicos

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`.

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`)

5.3.2 Números en coma flotante (`float`)

5.3.3 Operadores

Enteros (int)

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

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

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`.

Operadores aritméticos

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

Operadores de incremento/decremento

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

5.4. Cadenas (string)

5.4.1 Operadores de cadenas

5.4.2 Funciones de manejo de cadenas

5.4.3 Extensión *mbstring*

5.4. Cadenas (string)

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

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

Operadores de cadenas

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

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';`

Operador de incremento

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

Funciones de manejo de cadenas

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

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`)

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
- 6.2 Operadores de asignación compuesta
- 6.3 Comprobaciones
- 6.4 Conversiones de tipos
- 6.5 Comparaciones

6.1. Precedencia de operadores

6.1. Precedencia de operadores

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

6.2. Operadores de asignación compuesta

6.2. Operadores de asignación compuesta

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

6.3. Comprobaciones

6.3.1 De tipos

6.3.2 De valores

gettype()

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

`is_*()`

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

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

`is_numeric()`

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

ctype_*

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

6.4. Conversiones de tipos

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

6.4.2 Conversión a `bool`

6.4.3 Conversión a `int`

6.4.4 Conversión a `float`

6.4.5 Conversión de `string` a número

6.4.6 Conversión a `string`

6.4.7 Funciones de obtención de valores

6.4.8 Funciones de formateado numérico

6.4. Conversiones de tipos

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

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

Conversión a bool

http:
[//php.net/manual/es/language.types.boolean.php#language.types.boolean.casting](http://php.net/manual/es/language.types.boolean.php#language.types.boolean.casting)

Conversión a `int`

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

Conversión a float

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

Conversión de string a número

http:

[//php.net/manual/es/language.types.string.php#language.types.string.conversion](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`.

Conversión a string

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

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.

`intval()`

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

floatval()

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

strval()

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

boolval()

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

`number_format()`

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

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

6.5.2 `==` vs. `===`

6.5.3 Ternario (`?:`)

6.5.4 Fusión de `null` (`??`)

6.5.5 Reglas de comparación de tipos

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.

Ternario (?:)

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

Fusión de null (??)

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

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

Reglas de comparación de tipos

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

7. Constantes

7.1 Introducción

7.2 `define()` y `const`

7.3 Constantes predefinidas

7.4 `defined()`

7.1. Introducción

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

7.3. Constantes predefinidas

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

7.4. defined()

7.4. defined()

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

8. Ejercicios

8.1 Actividades

8.1. Actividades

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)?

9. Respuestas a las preguntas

9. Respuestas a las preguntas I

10. Bibliografía

10. Bibliografía