Conceptos básicos de PHP (I)

Ricardo Pérez López

IES Doñana, curso 2018-19

1. Introducción a PHP

1. Página web de PHP

http://php.net

1. Instalación de PHP

- Opción recomendada:

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

- Opción general:

  ```
  $ sudo apt install php
  ```

```
// Useless comment.
alert('hi');
```

When this fragment is shown, the first line of code (`span.line`) will have the `"focus"` class added to it.

Another fragment. This time, both lines will now have the `"focus"` class.

1. Documentación y búsqueda de información

- Manual en http://php.net/manual/es/
- Formulario de búsqueda en http://php.net

2. Sintaxis básica

2. Datos e instrucciones

123

2. Sentencias y comandos

123

2.2.. Comando `echo`

123

2. Expresiones, operadores y funciones

Ejemplos: aritmética, `cos()`, `max()`.

3. Funcionamiento del intérprete

3.1.. `php -a`

3.1.. PsySH

3. Modo dual de operación

Se llaman *modo HTML* y *modo PHP*.

4. Variables

4. Operadores de asignación por valor y por referencia

[link: En `$b =& $a;`, `$b` **NO** está apuntando a `$a` o viceversa. Ambos apuntan al mismo lugar.|http://php.net/manual/es/language.references.whatdo.php]

4. Variables predefinidas

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

5. Tipos básicos de datos

5. Lógicos (bool)

[link: Se escriben en minúscula: `false` y `true`.|https://github.com/yiisoft/yii2/blob/master/docs/internals/core-code-style.md#51-types]

`boolean` es sinónimo de `bool`, pero debería usarse `bool`.

5.1.. Operadores lógicos

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 [link: manual de PHP|http://php.net/manual/es/language.operators.precedence.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."* - [link: Pregunta que hice al respecto en StackOverflow|https://stackoverflow.com/questions/46861563/false-and-true-printhi].

5.2.. Enteros (`int`)

`integer` es sinónimo de `int`, pero debería usarse `int`.

5.2.. Números en coma flotante (`float`)

`double` es sinónimo de `float`, pero debería usarse `float`.

5.2.. Operadores aritméticos

5.2.. Operadores de incremento/decremento

5.3.. Concatenación

5.3.. Acceso y modificación por caracteres

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

5.3.. Operador de incremento #opcional

5. Nulo (null)

`is_null()` vs. `=== null`

[link: El tipo `null` y el valor `null` se escriben en minúscula.|https://github.com/yiisoft/yii2/blob/master/docs/internals/core-code-style.md#51-types]

6. Manipulación de datos

6. Operadores de asignación compuesta

```
$x <op>= $y
```

6.3.. `gettype()`

6.3.. `is_*()`

(poco útiles en formularios, ya que sólo se reciben `strings`)

6.3.. `is_numeric()`

6.3.. `ctype_*()`

6.4.. Conversión de `string` a número

¡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.. Funciones de obtención de valores

(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.. `intval()`

6.4.. `floatval()`

6.4.. `strval()`

6.4.. `boolval()`

6.4.. number_format()

6.4.. money_format()

set_locale()

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

6.5.. Operadores de comparación

`"250" < "27"` devuelve `false` 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.. Operador de coalescencia o fusión de null (??)

Equivalente al `COALESCE()` de SQL.

7. Constantes

7. Constantes

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.

8. Ejercicios