Funciones
en lenguaje PHP

Funciones predefinidas

- is_null($var): TRUE si es NULL.
- isset($var): TRUE si ha sido inicializada y no es NULL.
- unset($var): Elimina la variable (ya no estará inicializada).
- empty($var): TRUE si no ha sido inicializada o su valor es FALSE.
- is_int($var), is_float($var), is_bool($var), is_array($var).
- print_r($var): Muestra información sobre la variable.

Funciones propias

- Para crear tus propias funciones, deberás usar la palabra **function**:

```
function suma($a, $b){
    return $a + $b;
}
```

- Para invocar la función:

```
$resultado = suma(5, 7);
```

- Si una función no tiene una sentencia **return**, devuelve null al finalizar.

Valores por defecto en los parámetros

- Podemos indicar valores por defecto para los parámetros.
- Si cuando llamamos a la función no indicamos el valor de un parámetro se tomará el valor por defecto indicado.

```
function precio_con_iva($precio, $iva=0.21){
    return $precio * (1 + $iva);
}
$precio = 10;
$precio_iva = precio_con_iva($precio);
```

- Puede haber más de un parámetro con valor por defecto, pero **siempre tienen que estar al final**.

Paso de parámetros por referencia

- Por defecto los parámetros se pasan por valor.
- Para pasar un parámetro por referencia añadiremos el símbolo **&** delante de su nombre.

```
function precio_con_iva(&$precio, $iva=0.18){
    $precio *= (1 + $iva);
}
```

Declaraciones de tipo

- Las funciones obligan a que los parámetros sean de cierto tipo.
- Si el valor dado es de un tipo incorrecto, se generará un error.
- Debe anteponerse el nombre del tipo al nombre del parámetro.
- Se puede hacer que una declaración acepte valores NULL si el valor predeterminado del parámetro se establece a NULL.

```
function suma(int $a, int $b){
    return $a + $b;
}
$resultado = suma(5, 3);
```

Tipos válidos

Tipo	Descripción	Versión de PHP mínima
nombre de clase/interfaz	El parámetro debe ser una *instanceof* del nombre de la clase o interfaz dada.	PHP 5.0.0
self	El parámetro debe ser una *instanceof* de la misma clase donde está definido el método. Esto solamente se puede utilizar en clases y métodos de instancia.	PHP 5.0.0
array	El parámetro debe ser un array.	PHP 5.1.0
callable	El parámetro debe ser un callable válido.	PHP 5.4.0
bool	El parámetro debe ser un valor de tipo boolean.	PHP 7.0.0
float	El parámetro debe ser un número de tipo float.	PHP 7.0.0
int	El parámetro debe ser un valor de tipo integer.	PHP 7.0.0
string	El parámetro debe ser un string.	PHP 7.0.0

Declaraciones de tipo de devolución

- Añadido en PHP 7.
- Especifican el tipo del valor que serán devuelto desde una función.
- Están disponibles los mismos tipos de la tabla anterior.

```php
function suma(int $a, int $b) : int {
    return $a + $b;
}
$resultado = suma('5', 3);
```

Tipificación estricta

- PHP fuerza a los valores de un tipo erróneo a ser del tipo escalar esperado si es posible.
  - Ejemplo: Una función que espera un string y recibe un int obtendrá una variable de tipo string.
- En el modo estricto esto no se permite.
- Si los tipos no coinciden nos dará error.
- **Excepción**: se puede pasar un int a una función que espere un float.
- El modo estricto se habilita para cada fichero php:

```
declare(strict_types=true);
```

Funciones anónimas

- Están implementadas usando la clase Closure.
- Permiten la creación de funciones que no tienen un nombre específico.
- Podemos asignar la función a una variable o pasarla como parámetro a otra función.

Funciones anónimas

- Sin parámetros:

```php
$anonima = function () {
    echo "Hola";
};
$anonima();
```

- Con parámetros:

```php
$anonima = function ($nombre) {
    echo "Hola {$nombre}";
};
$anonima('Álex');
```

Usar variables del ámbito superior

- Una función anónima puede usar variables del ámbito superior mediante la palabra reservada **use**:

```
function saluda(callable $fnSaluda) {
    $fnSaluda('Álex');
}
$saludo = 'Hola';
$anonima = function ($nombre) use ($saludo) {
    echo "{$saludo} {$nombre}";
};
saluda($anonima);
```