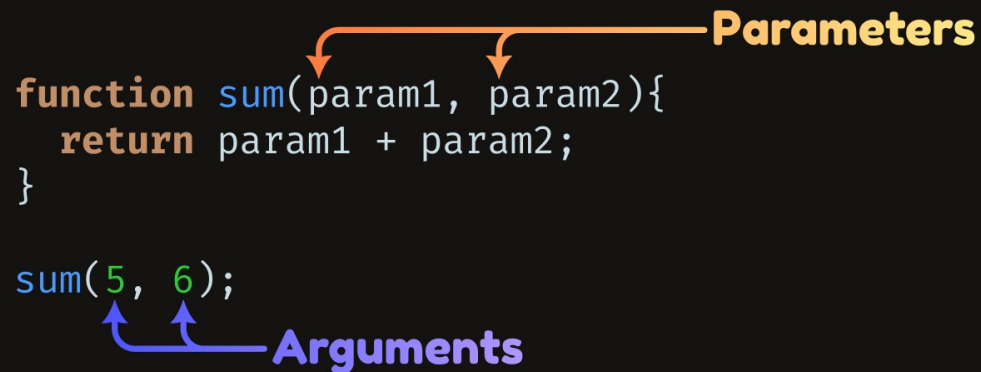


PHP 11



The diagram illustrates the relationship between function parameters and arguments. It shows a function definition `function sum(param1, param2){` and a function call `sum(5, 6);`. An orange bracket labeled "Parameters" points to `param1` and `param2` in the function definition. A blue bracket labeled "Arguments" points to `5` and `6` in the function call.

```
function sum(param1, param2){  
    return param1 + param2;  
}  
  
sum(5, 6);
```

Funcions: Arguments

15)Funcions: Arguments

- Qualsevol **informació pot ser passada** a les **funcions** mitjançant la **llista d'arguments o paràmetres**: llista d'expressions **delimitades per comes**.
- Els arguments són **avaluats d'esquerra a dreta**.
- PHP admet:
 - **Pas** d'arguments **per valor** (per defecte).
 - **Pas** de paràmetres **per referència**.
 - **Valors** d'arguments **predeterminats**.

Pas d'arguments per valor

- **Per defecte en PHP**, els arguments de les funcions són **passats per valor**:
 - Si el **valor** de l'argument **canvia dins de la funció**:
 - Aquest **no canvia fora de la funció**.
 - El **canvi** d'una **dada d'un paràmetre dins d'una funció**:
 - **No actualitza la dada** de la **variable** que es va **passar a la funció**.

Exemple 1

```
<?php
function per_valor($parametre1) {
    //modifiquem el valor de la variable de l'argument definit a la funció
    $parametre1 = "hola";
    echo $parametre1."<br/>"; //Imprimeix "hola"
}
```

```
$variable = "açò no canvia";
per_valor($variable);
echo $variable."<br/>"; //Imprimeix "açò no canvia"
?>
```

• Resultat:
hola
açò no canvia

Pas d'arguments per referència

- Per **permetre** a una **funció modificar** els seus **arguments**:
 - Han de **passar-se per referència**.
- Perquè un **argument** a una funció siga sempre **passat per referència en PHP**:
 - **Anteposar** al **nom de l'argument** el símbol '**&**' en la **definició de la funció**.

Exemple 2

<?php //Pas de paràmetres d'una funció per referència

```
function afegir_alguna_cosa (&$cadena) { // símbol & davant de l'argument
```

```
    $cadena = $cadena.' i alguna cosa més.';
```

```
}
```

```
$cad ='Això és una cadena,';
```

```
afegir_alguna_cosa($cad);
```

```
//el valor de la variable $cad ha sigut modificat en la funció
```

```
echo $cad;
```

```
?>
```

- Resultat:

Això és una cadena, i alguna cosa més.

Valors d'arguments predeterminats

- Una funció pot definir **valors predeterminats** per als seus **arguments**.
- El **valor per defecte** ha de ser una **expressió constant**:
 - **No pot ser**, per exemple:
 - Una **variable**.
 - Un **membre d'una classe**.
 - Una **crida a una funció**.
 - **Sí que pot ser**, per exemple, a part d'un **valor literal o constant**:
 - **Array**.
 - **null**.

Exemple 3

<?php //Ús de paràmetres predeterminats en funcions

```
function fer_cafe ($tipus = "capucino") { // símbol = per al valor per defecte
    return "Fer una tassa de cafè $tipus. <br/>";
}
echo fer_cafe();
echo fer_cafe(null);
echo fer_cafe("espresso");
?>
```

- Resultat:

Fer una tassa de capucino.
Fer una tassa de.
Fer una tassa de espresso.

- Podem definir el **valor per defecte** de **diversos arguments en una mateixa funció**.
- Per **exemple**:

Valor per defecte de
\$parametre1 és "David"

Valor per defecte de
\$parametre2 és 3

```
function funcio1($parametre1 = "David", $parametre2 = 3){...}
```

- Crida 1: `funcio1()` `//$parametre1 = "David", $parametre2 = 3`
- Crida 2: `funcio1("hola")` `//$parametre1 = "hola", $parametre2 = 3`
- Crida 3: `funcio1(5)` `//$parametre1 = 5, $parametre2 = 3`
- Crida 4: `funcio1("col",6)` `//$parametre1 = "hola", $parametre2 = 6`

- Per utilitzar **paràmetres amb valors per defecte**:
 - S'han de **declarar obligatòriament al final en la llista de paràmetres** de la interfície (capçalera) de la funció.
 - Els paràmetres que **no tinguen valors per defecte**:
 - Es **declaran a l'inici**.
- Quan s'utilitzen **arguments predeterminats**:
 - Qualsevol d'ells hauria d'estar a la **dreta dels arguments no predeterminats**:
 - Si no, les coses **no funcionaran com s'esperava**.

Exemple 4

<?php // Ús incorrecte d'arguments predeterminats en una funció

```
function fer_iogurt($tipus = "ensucrat", $sabor) {  
    return "Fer un bol de iogurt $tipus de $sabor. <br/>";  
}  
echo fer_iogurt("plàtan"); // No funcionarà com s'esperava  
?>
```

- Resultat:

Warning: Missing argument 2 for fer_iogurt()...

Exemple 5

<?php // Ús correcte d'arguments predeterminats en una funció

```
function fer_iogurt($sabor, $tipus = "ensucrat") {  
    return "Fer un bol de iogurt $tipus de $sabor.<br/>";  
}
```

```
echo fer_iogurt("plàtan"); // Funciona com s'esperava
```

?>

- Resultat:

Fer un bol de iogurt ensucrat de plàtan.

Declaració de tipus (Determinació de tipus)

- Permet a les funcions **requerir** que els **paràmetres** siguin **de cert tipus** en una **crida**.
 - Si el valor donat és d'un **tipus incorrecte**: es generarà un **error**.
 - Funcionalitat vàlida a partir de **php 7** (per a la majoria dels tipus).

- **Declaració de tipus**: **anteposar el nom del tipus** al **nom del paràmetre**:

```
function funcio2(string $cad1, float $num){...}
```

- Es pot fer que una declaració accepti **valors null** si el **valor predeterminat** del paràmetre s'estableix a **null**.

Exemple 6

```
<?php
```

```
function sumaEnters(int $a, int $b) {  
    return $a + $b;  
}  
  
echo "Suma: ".sumaEnters(3,7)."<br/>"; //crida 1  
echo "Suma: ".sumaEnters("Tres",5)."<br/>"; //crida 2
```

```
?>
```

- Resultat:

Suma: 10

Fatal error: Uncaught TypeError: sumaEnters(): Argument #1 (\$a) must be of type int, string given

Valor de retorn

- Es pot indicar el **tipus de dades de retorn** de les **funcions**:
 - Es fica **a continuació de la capçalera** de la funció, **precedit pels dos punts (:)**.
- Vàlid a partir de la versió **PHP 7**.
- Els **tipus que accepta PHP** com a retorn de funcions són els següents:
 - int
 - float
 - bool
 - string
 - array
 - interfaces
 - callable

- Indicar que PHP **no faça conversions entre tipus** per adequar els valors de les variables:
 - Establir el **mode de treball** `strict_types`.
- Per **Exemple**:

```
<?php
```

```
declare(strict_types = 1); // els tipus esperats són els declarats
function mostraInt(int $valor): int { // hem de tornar un int
    return $valor;
}
echo "Valor: ".mostraInt(5);

?>
```

- Resultat:

Valor: 5

PHP 11: Funcions: Arguments

- Si intentem retornar un valor que no fora int, estant activat el **mode de treball**

strict_types: donaria un **error**.

- Per **Exemple**:

```
<?php
declare(strict_types = 1); // els tipus esperats són els declarats
function mostraInt2(int $valor): int { // hem de retornar int
    return $valor + 5.0; // intentem tornar un float
}
echo "Resultat: ".mostraInt2(5);
?>
```

- Resultat:

Fatal error: Uncaught TypeError: mostraInt2(): Return value must be of type int, float returned

- Si intentem retornar un valor que no fora int, estant activat el **mode de treball**

strict_types: donaria un **error**.

- Per **Exemple**:

```
<?php
// declare(strict_types = 1); // ara no activem el mode strict_types
function mostraInt2(int $valor): int { // hem de retornar int
    return $valor + 5.0; // intentem tornar un float
}
echo "Valor: ".mostraInt2(5);
?>
```

- Resultat:

Valor: 10