

# Desarrollo de sitios web con PHP y MySQL



## Tema 3: Funciones

Ana M<sup>a</sup> Feroso García  
afermosoga@upsa.es

# Tema 3: Funciones

1. Definición de funciones de usuario. Paso de parámetros
2. Return
3. Funciones variable
4. Bibliotecas de funciones:
  1. Funciones para fechas
  2. Funciones para cadenas

# Definición de Funciones

- Sintaxis

```
function nom_funcion ([&]$param1, [&]$param2, ...)  
{  
    sentencias  
    [return valor;]  
}
```

- Ejemplo:

```
function suma ($x, $y)  
{  
    $s = $x + $y;  
    return $s;  
}
```

```
$a=1;  
$b=2;  
$c=suma ($a, $b);  
print $c;
```

# Paso de parámetros (I)

- Por defecto los parámetros se pasan **por valor**
- Paso **por referencia**: `&$nom_parametro`

```
function incrementa (&$a)
{
    $a = $a + 1;
}
```

```
$a=1;
incrementa ($a);
print $a; // Muestra un 2
```

# Paso de parámetros (II)

- Argumentos **por defecto**

```
function muestranombre ($titulo = "Sr.")  
{  
    print "Estimado $titulo:\n";  
}  
muestranombre ();  
muestranombre ("Prof.");
```

- Salida:

```
Estimado Sr.:  
Estimado Prof.:
```

# Paso de parámetros (III)

- Los argumentos con valores por defecto deben ser siempre los últimos:

```
function muestranombre ($nombre, $titulo= "Sr.")  
{  
    print "Estimado $titulo $nombre:\n";  
}  
muestranombre ("Fernández");  
muestranombre ("Fernández", "Prof.");
```

- Salida:

```
Estimado Sr. Fernández:  
Estimado Prof. Fernández:
```

# Sentencia `return`

- **Sintaxis:** `return [valor];`
  - Pone fin a la ejecución y puede devolver algún valor
  - Ejemplo:

```
function máximo($x, $y)
{
    if ((isset($x) || isset($y))
        return false
    else if ($x >= $y)
        return $x
    else
return $y
```

# Funciones variable

- Si una variable tiene unos paréntesis añadidos a l final, PHP buscará una función con el mismo nombre que el contenido de la variable e intentará ejecutarla
- Ejemplo

```
$mifunc="máximo";
```

```
$mifunc(3,7); //llama y ejecuta la función: máximo(3,7)
```



# Bibliotecas de funciones

- Existen muchas bibliotecas de funciones en PHP
- Algunos ejemplos:
  - Funciones de manipulación de cadenas
  - Funciones de fecha y hora
  - Funciones de arrays
  - Funciones de ficheros
  - Funciones matemáticas
  - Funciones de bases de datos
  - Funciones de red
- Algunas bibliotecas requieren la instalación de componentes adicionales
- Todas las funciones de biblioteca están comentadas en la documentación de PHP

# Funciones con cadenas

- Combinar y dividir cadenas
  - **explode()**
    - Divide una cadena en subcadenas
    - array **explode** (string separator, string string [, int limit])
  - **implode()**
    - Junta un array de cadenas en una sola separadas por un separador
    - string **implode** (string separator, array\_cadenas)
  - **substr()**
    - Obtener subcadena de cadena
    - string **explode** (string cadena, int inicio, [int longitud])
- Limpiar cadenas
  - **rtrim(), ltrim(), trim()**
    - Eliminan caracteres a la derecha, a la izquierda o por ambos lados de una cadena
    - string **rtrim** ( string str [, string charlist])
- Aplicar formato
  - **addslashes()**
    - Marcar caracteres especiles (" , ' \ y NULL) con \ antes de almacenar en BD
    - string **addslashes** ( string cadena)
  - **nl2br()**
    - Añade <br/> al final de cada linea en cadenas largas
    - string **nl2br** ( string cadena)
  - **strtolower(cadena) / strtoupper(cadena)**
    - Convierte una cadena a minúscula / mayúscula
- Buscar y reemplazar / comparar / longitud
  - **strstr() / strchr()**
    - Busca la primera ocurrencia de una subcadena (o un carcater) y devuelve subcadena partir de coincidencia
    - string **strstr**(cadena, cadena\_busca)
    - string **strstr**(cadena, caracter\_busco)
  - **strcmp(cadena1, cadena2) / strcasecmp(cadena1, cadena2)**
    - Compara dos cadenas con/sin distinción de mayúsculas
  - **strlen(cadena)**
    - Calcula la longitud de una cadena

# Funciones de fecha y hora (I)

- **time()**: tiempo en segundos. Formato timestamp
- **date(formato, timestamp)**: formatea fecha actual o la que se le pase como segundo parámetro
  - Códigos formato
    - a : am o pm
    - A : AM o PM
    - D : día de mes con ceros
    - F : nombre del mes en inglés
    - h: hora formato 1-12
    - H: hora formato 0-23
    - i: minutos
    - J: día del mes sin ceros
    - L: día de la semana
    - m: número del mes (1-12)
    - M: abreviatura del mes (ing)
    - s: segundos
    - y: año con 2 dígitos
    - Y: año con 4 dígitos
    - z: día del año (1-365)
  - Ejemplo. Si hoy es 12 de octubre de 2011 y son las 15:20:05
    - date ("d-m-y H:i:s") imprime 12-10-2011 15:20:05

# Funciones fecha y hora (II)

- **mktime(hora, min, seg, mes, día, año):** convierte fecha a timestamp (sobre todo para operaciones entre fechas)

- **Ejemplo:**

```
$fecha = date ("j/n/Y", mktime(3,20,50,4,5,2001));  
print ("$fecha");
```

*Resultado:* 5/4/2001

- **checkdate(mes,día, año):** comprueba si una fecha es válida y devuelve true o false

# Expresiones regulares

- **preg\_match**(string patron, string cadena, [matriz\_coincidencias], [modificador(PREG\_OFFSET\_CAPTURE)], [int desplazamiento])

Compara una cadena con un patrón y devuelve 1 si el patrón ha coincidido y 0 si no es así. Cada coincidencia encontrada se puede guardar opcionalmente en `matriz_coincidencias`, y con el modificador `PREG_OFFSET_CAPTURE`, se guarda también en la matriz la posición en que se ha encontrado cada coincidencia. “desplazamiento” es un número que indica en qué carácter se inicia la búsqueda.

- **Sintaxis:**
  - **Patrón:** debe empezar y acabar con el carácter /
  - Se distingue entre mayúsculas y minúsculas. Para evitarlo hay que añadir el modificador “i” (sin comillas) al final del patrón después de la \, aunque esto no afecta a las clases `[::]`
  - Clases de carácter : `[:alnum/alpha/digit/lower/...:]`

# Sintaxis patrones

Patrón	Significado
<code>c</code>	carácter <code>c</code>
<code>.</code>	cualquier carácter
<code>^c</code>	empezar por el carácter <code>c</code>
<code>c\$</code>	terminar por el carácter <code>c</code>
<code>c+</code>	1 o más caracteres <code>c</code>
<code>c*</code>	0 o más caracteres <code>c</code>
<code>c?</code>	0 o 1 caracteres <code>c</code>
<code>\n</code>	nueva línea
<code>\t</code>	tabulador
<code>\</code>	escape, para escribir delante de caracteres especiales: <code>^ . [ ] % ( )   * ? { } \</code>
<code>(cd)</code>	caracteres <code>c</code> y <code>d</code> agrupados
<code>c d</code>	carácter <code>c</code> o <code>d</code>
<code>c{n}</code>	<code>n</code> veces el carácter <code>c</code>
<code>c{n,}</code>	<code>n</code> o más caracteres <code>c</code>
<code>c{n,m}</code>	desde <code>n</code> hasta <code>m</code> caracteres <code>c</code>
<code>[a-z]</code>	cualquier letra minúscula
<code>[A-Z]</code>	cualquier letra mayúscula
<code>[0-9]</code>	cualquier dígito
<code>[cde]</code>	cualquiera de los caracteres <code>c</code> , <code>d</code> o <code>e</code>
<code>[c-f]</code>	cualquier letra entre <code>c</code> y <code>f</code> (es decir, <code>c</code> , <code>d</code> , <code>e</code> o <code>f</code> )
<code>[^c]</code>	que no esté el carácter <code>c</code>
<code>[:alnum:]</code>	cualquier letra o dígito
<code>[:alpha:]</code>	cualquier letra
<code>[:digit:]</code>	cualquier dígito
<code>[:lower:]</code>	cualquier letra minúscula
<code>[:punct:]</code>	cualquier marca de puntuación
<code>[:space:]</code>	cualquier espacio en blanco
<code>[:upper:]</code>	cualquier letra mayúscula