


PHP

Prof. Dely Maybel Gil A.



Poned, pues, ahora vuestros corazones y vuestros ánimos en buscar a vuestro Dios; y levantaos, y edificad el santuario de Dios, para traer el arca del pacto del Señor, y los utensilios consagrados a Dios, a la casa edificada al nombre del Señor.

1 Crónicas 22:19



Contenido



1

¿Qué es PHP?

2

Introducción a PHP

3

Sentencias de Control

4

Funciones

5

Cadenas de Caracteres

6

Arreglos

- ❖ PHP Hypertext Pre-Processor
- ❖ Lenguaje de programación Interpretado
- ❖ Ejecutable del lado del servidor
- ❖ Heredero de C++
- ❖ Embebido en HTML
- ❖ De propósito general
- ❖ Orientado a programación Web
- ❖ Software Libre
- ❖ Amplio soporte de base de datos: Adabas D, dBase, Empress, Ingress, InterBase, FrontBase, DB2, Informix, mSQL, MySQL, ODBC, Oracle, PostgreSQL, Sybase

Verificar Instalación de PHP

```
<?php  
    phpinfo();  
?>
```



Sintaxis de PHP



- ❖ `<?php //Comienzo de código PHP`
- ❖ `//FORMATO 1`
- ❖ `echo "<p><hr>";`
- ❖ `echo "La fecha actual es: ";`
- ❖ `?> //Fin de código PHP`
- ❖ `<?`
- ❖ `//FORMATO 2`
- ❖ `echo gmDate("d,m,Y.");`
- ❖ `echo "
";`
- ❖ `?>`
- ❖ `<script language="php">`
- ❖ `//FORMATO 3`
- ❖ `echo gmDate("h:i:s A");`
- ❖ `</script>`
- ❖ `<?php`
- ❖ `// Barra invertida: sustituye las dobles comillas`
- ❖ `echo "<p> !PHP es una \"bomba\"!</p> ";`
- ❖ `?>`

00Sintaxis.php



Variables en PHP



- ❖ No tienen tipo de datos definido
- ❖ Aunque no se le asigna tipo de dato, la variable tomará el tipo de dato que contenga el valor que se le asigne.
- ❖ No se declaran
- ❖ Empiezan en \$
 - Ejemplo: \$nombre, \$edad
 - \$nombre = "Josué"
 - \$edad = 26
- ❖ No pueden empezar con números ni caracteres especiales.
- ❖ Tipos de Datos
 - Integer Double String
 - Boolean Array Object



Tipos de Datos



01TipoDato.php

- ❖ `<?php`
- ❖ `//tipo de dato Integer`
- ❖ `$a = 1234;`
- ❖ `$b = -1234;`
- ❖ `echo "$a, $b";`
- ❖ `$c = 0123; # OCTAL -->83 decimal`
- ❖ `$d = 0x12; # HEXA -->18`
- ❖ `echo "
".$c. " ". $d;`
- ❖ `//tipo de dato double`
- ❖ `$a = 1.2e3;`
- ❖ `echo "
 $a", "
";`
- ❖ `//tipo de dato CADENA`
- ❖ `$a = "Mundo";`
- ❖ `echo "hola $a
";`
- ❖ `echo 'Hola $a'. "
"; // entre simple comillas no imprime el valor de $a`

Funciones de utilidades para variables

- ❖ **gettype (\$var)** : Retorna el tipo de dato de una variable
- ❖ **settype(\$var, tipodato)**: Cambia el tipo de dato de una variable.
Los tipos de datos posibles son: integer, double, string, array, object
- ❖ **isset(\$var)**: Determina si una variable está definida (true,false)
- ❖ **empty (\$var)** : Determina si una variable está definida
- ❖ **is_tipo_dato(\$var)** : Averigua si una variable es del tipo_dato
- ❖ **tipo_datoval (\$var)**: Convierte la variable al tipo de dato
(int,double,str)

Funciones de utilidades para variables

- ❖ //gettype: Retorna el tipo de dato de una variable
- ❖ \$edad = 3;
- ❖ \$nombre = "Maria";
- ❖ print (gettype(\$edad)."
");
- ❖ print (gettype(\$nombre)."
");
- ❖ //settype(): Cambia el tipo de dato de una variable
- ❖ \$var = 7.8;
- ❖ settype(\$var,"integer"); echo \$var;
- ❖ //isset : Determina si una variable está definida (true 1,false 0)
- ❖ echo "a: ".isset(\$a)."
 edad: ".isset(\$edad);
- ❖ //empty : Determina si una variable está definida
- ❖ print (empty(\$var));
- ❖ \$var = 1;
- ❖ print ("
".empty(\$var)."hhhh
");

01TipoDato.php

Funciones de utilidades para variables

- ❖ **//IS: Averigua si una variable es <tipo_dato>**
- ❖ \$var = "Cristiano";
- ❖ if (is_integer(\$var)){
- ❖ print("Integer");
- ❖ }elseif(is_double(\$var)){
- ❖ print("Double");
- ❖ }elseif(is_string(\$var)){
- ❖ print("String");
- ❖ }
- ❖ **// ...VAL : Convierte la variable al tipo de dato (int,double,str)**
- ❖ echo "
";
- ❖ \$var = "28020IUTVAL";
- ❖ \$varM = intval(\$var);
- ❖ echo \$varM;

01TipoDato.php

[Regresar](#)



Constantes



- ❖ Para crear constantes se utiliza la función:

- `define(nombre constante, valor);`

01TipoDato.php

- ❖ **// definir constantes al principio!!!!**

- ❖ `define("CIUDAD", "MERIDA");`

- ❖ `define("COLOR", "#3366FF");`

- ❖ `define("PAGINAS", 4);`

- ❖ `if (defined("COLOR")){ //comprueba si una constante ha sido creada`

- ❖ `//print(CIUDAD);`

- ❖ `print("<p>");`

- ❖ `echo CIUDAD; print("</p>");}`

- ❖ `?> //Final de código PHP 01TipoDato.php`

❖ Aritméticos

- = (asignación)
- + (suma)
- - (resta)
- * (multiplicación)
- / (división)
- % (resto de la división entera)
- Operador= por ejemplo
\$a+=\$b es equivalente a
\$a=\$a+\$b
- Variable++ (incremento)
- Variable-- (decremento)

❖ Comparativos

- == (igualdad en valor)
- < (menor que)
- > (mayorque)
- <= (menor o igual que)
- >= (mayor o igual que)
- != (distinto de)
- === (comparación de valor y tipo)

❖ Lógicos

- ! (negación)
- && (and)
- || (or)

❖ Cadenas

- . (concatenación)



Sentencia if else



02EstructurasControl.php

```
❖ <?php
❖     $x=3;
❖     $y=2;
❖     if ($x == $y)
❖     { print ("tanto y como x son iguales"); }
❖     else
❖     { print ("son diferentes"); }
❖ ?>
```

Sentencia if –elseif

- [Ver ejemplo de la función IS](#)

- ❖ **Operador ternario**
- ❖ `($x == $y)? print "Son iguales" : print "Son diferentes";`
- ❖ **Sintaxis Alternativa**
 - `if():`
 - `.... endif`



Sentencia switch



02EstructurasControl.php

```
❖ $a=3;
❖ switch($a) {
❖     case 1:
❖         print("estamos en la opción uno");
❖         break;
❖     case 2:
❖         print("estamos en la opción dos");
❖         break;
❖     case 3:
❖         print("estamos en la opción tres");
❖         $a--;
❖         break;
❖     default:
❖         print("No hay opciones");
❖ }
❖ print("Valor de a: " . $a."<br>");?>
```

❖ */* ejemplo 1 */*

❖ `for ($i = 1; $i <= 10; $i++) { print $i; }`

02EstructurasControl.php

❖ */* ejemplo 2 */*

❖ `for ($i = 1; ; $i++) {`

❖ `if ($i > 10) {`

❖ `break;`

❖ `}`

❖ `print $i;`

❖ `}`

❖ */* ejemplo 3 */*

❖ `for ($i = 1; $i <= 10; print $i, $i++) ;`

/ No necesariamente expr2 tiene porque ser siempre numérica*

`for ($i=0;checkEmail()!="error";$i++){`

Ejecutar hasta que la función

checkEmail devuelva error */

❖ */* ejemplo 4 */*

❖ `//sentencia alternativa`

❖ `for ($i=0;$i<10;$i++):`

❖ `print("línea..".$i."
")`

❖ `;`

❖ `endfor;`

- ❖ `/* ejemplo 1 */`
- ❖ `$var = 10;`
- ❖ `while ($var >0){`
- ❖ `print ("Nuevo valor: ".$var."
");`
- ❖ `--$var;`
- ❖ `}`
- ❖ `/*ejemplo 2 */`
- ❖ `$var = 8;`
- ❖ `while (--$var)`
- ❖ `print ("Valor: ".$var."
");`
- ❖ `/*ejemplo 3 */`
- ❖ `$var = 12;`
- ❖ `do{`
- ❖ `print("el valor es: ".$var."
");`
- ❖ `--$var;`
- ❖ `} while ($var>8);`

Incluir archivos, funciones, o partes de código (su uso más común, para agregar la cabecera y pie de una página).

SINTAXIS:

`<? include("archivo.php") ?>` :

- ❖ Las funciones y variables definidos antes de la llamadas son accesible para el código en el fichero. De igual forma todos los elementos definidos en el fichero estarán disponibles par el script llamante.
- ❖ Si el fichero a incluir no existe se genera un warning o un aviso, continuando con la ejecución por la siguiente instrucción
- ❖ Puede utilizarse en combinación con otras estructuras de control

```
for ($i;$i<4;$i++) {include("fichero".$i.".php");}
```

SINTAXIS:

`<? require("archivo") ?>` : Incluye todo lo contenido

- ❖ Sólo incluye el fichero referenciado, es decir, no lo interpreta.
- ❖ Su comportamiento es equivalente a la directiva `#include` del Leng. C
- ❖ No puede ser utilizado con estructuras de control
- ❖ En caso que el archivo referenciado por `require` no existe, se genera un error fatal y no permite seguir ejecutando el script

`require_once()` o `include_once()`:

- ❖ Carga y evalúa cada script una vez como máximo, evitando así los errores producidos por redefinición de funciones o la resignación de valores a variables
- ❖ La diferencia entre ambos es la misma que `include()` y `require()`.



Inclusión de Archivos



```
<h3>cabecera</h3>
```

```
cabecera.inc
```

```
<?php
    $resultado=1;
    for($i=1;$i<=abs($exponente);$i++){
        $resultado*=$base;
    }
    if ($exponente<0){
        $resultado=1/$resultado;
    }
    echo "$base<SUP>$exponente</SUP> = ";
    echo "<B>$resultado</B>";
?>
```

```
cuerpo.inc
```

```
<br><h3>Último resultado...<?=$resultado?></h3>
```

```
pie.inc
```



Inclusión de Archivos



```
<html><head><title>Trabajando con INCLUDE</title></head>
<body><center>
  <h2>Inclusión de ficheros </h2>
  <?php
    //se incluye la cabecera html
    include("cabecera.inc");
    //el fichero "cuerpo.inc hace uso de las variables $base y $exponente
    $base = 2;    $exponente= -3;
    include("cuerpo.inc");
    echo "<br>Primera inclusion de 'cuerpo.inc<hr width='40%'>";

    $base = 7;    $exponente= 5;
    include("cuerpo.inc");
    echo "<br>Segunda inclusion de 'cuerpo.inc<hr width='40%'>";

    $base = 9;    $exponente= 7;
    include("cuerpo.inc");
    echo "<br>Tercera inclusion de 'cuerpo.inc<hr width='40%'>";

    //se incluye un fichero que hace uso de variables definidas en 'cuerpo.inc'
    include("pie.inc");
  ?>
</center></body></html>
```

Inclusion.php

Inclusión de ficheros

cabecera

$$2^{-3} = 0.125$$

Primera inclusion de 'cuerpo.inc'

$$7^5 = 16807$$

Segunda inclusion de 'cuerpo.inc'

$$9^7 = 4782969$$

Tercera inclusion de 'cuerpo.inc'

Último resultado...4782969



Inclusión de Archivos



```
<html><head><title>Trabajando con INCLUDE</title></head>
<body><center>
  <h2>Inclusión de ficheros </h2>
  <?php
    //se incluye la cabecera html
    include_once("cabecera.inc");
    //el fichero "cuerpo.inc hace uso de las variables $base y $exponente
    $base = 2;    $exponente= -3;
    include_once("cuerpo.inc");
    echo "<br>Primera inclusion de 'cuerpo.inc<hr width='40%'>";

    $base = 7;    $exponente= 5;
    include_once("cuerpo.inc");
    echo "<br>Segunda inclusion de 'cuerpo.inc<hr width='40%'>";

    $base = 9;    $exponente= 7;
    include_once("cuerpo.inc");
    echo "<br>Tercera inclusion de 'cuerpo.inc<hr width='40%'>";

    //se incluye un fichero que hace uso de variables definidas en 'cuerpo.inc'
    include_once("pie.inc");
  ?>
</center></body></html>
```

Inclusion_once.php

Inclusión de ficheros

cabecera

$$2^{-3} = 0.125$$

Primera inclusion de 'cuerpo.inc'

Segunda inclusion de 'cuerpo.inc'

Tercera inclusion de 'cuerpo.inc'

Último resultado...0.125



Funciones



- ❖ Función con parámetro por valor:
function nombre (\$par).
- ❖ Función con parámetro por referencia:
function nombre (&\$par).
- ❖ Función con parámetro con valor por defecto:
function nombre (\$par = valor).

Nota:

- ❖ Para retornar el valor de una función se usa return.
- ❖ El PHP permite recursividad en sus funciones.

- ❖ Variables del tipo static(static\$var)
siempre retorna el último valor que tuvo esa variable.
- ❖ Variables del tipo global (global \$var)
- ❖ Obtener número de parámetros enviados / contenido de parámetros:
func_nums_args() / func_get_arg(\$i) / arg_list[\$i]

```
❖ function no_arg(){ // Sin parámetros
❖     echo "Hola mundo Cristiano","<br>";
❖ }
❖ echo "No se ha llamado a la función NO_ARG todavía","<br>";
❖ no_arg();
❖ echo "La función NO__ARG ha sido llamada!","<br>";
```

03Funciones.php

```
❖ function contar($number=6){ //parámetros por defecto
❖     for(;$number<10;$number++){
❖         echo $number,"<br>";
❖     }
❖ }
❖ echo "No se ha llamado la función CONTAR todavía <br>";
❖ contar(8);
❖ contar();
❖ echo "!La función CONTAR ha sido llamada<br>";
```

- ❖ `function cuadrado($num){ // parámetro por valor, pasa una copia`
- ❖ `$num = $num + 1;`
- ❖ `return $num*$num;`
- ❖ `}`
- ❖ `$numero = 7;`
- ❖ `print($numero."
");`
- ❖ `print(cuadrado($numero."
");`
- ❖ `print($numero."
");`
- ❖ `//parámetro por referencia (& antes del argumento del parámetro)`
- ❖ `function anadir(&$string) {`
- ❖ `$string .= ' y algo más.';`
- ❖ `}`
- ❖ `$str = 'Esto es una cadena, ';`
- ❖ `anadir($str);`
- ❖ `echo $str."
"; // 'Esto es una cadena, y algo más.'`

03Funciones.php



Funciones



//Alcance de las Variables

04Funciones.php

```
❖ $numero = 9;
❖ function otro_numero(){
❖     $numero=6;
❖ }
❖ print("El número es :".$numero."<br>"); otro_numero();
❖ print("El valor de la variable".$numero." no se ve alterada <br>");
```

```
❖ $numero = 9;
❖ function otro_numero2(){
❖     global $numero;
❖     $numero=6;
❖ }
❖ print("El numero es :".$numero."<br>"); otro_numero2();
❖ print("El valor de la variable".$numero." si se ve alterada <br>");
```

```
❖ //Array $GLOBALS
❖ $numero = 9;
❖ function otroNum(){
❖     $GLOBALS["numero"]=6 ; // cuando se usa el array las variables no van precedidas $
❖ }
❖ otroNum();
❖ print("Número con GLOBALS: ".$numero."<br>");
```



Funciones



04Funciones.php

❖ //La sentencia static

❖ function contador(){

❖ \$contador=0;

❖ \$contador++;

❖ return \$contador;

❖ }

❖ print(contador()."
");

❖ print(contador()."
");

❖ print(contador()."
");

❖ //Usando static

❖ /*function contador(){

❖ static \$contador=0;

❖ \$contador++;

❖ return \$contador;

❖ }

❖ print(contador()."
");

❖ print(contador()."
");

❖ print(contador()."
");

❖ */



Funciones



//Lista de longitud variable de parámetros

- ❖ //int func_num_args(void)
- ❖ echo "func_num_args
";
- ❖ **function facil(){**
- ❖ \$numargs = func_num_args();
- ❖ echo "Número de argumentos en facil: \$numargs
";
- ❖ }
- ❖ **facil(1,2,3);**

- ❖ //int func_get_arg(int arg_num)
- ❖ echo "func_get_arg
";
- ❖ **function facilit() {**
- ❖ \$numargs = func_num_args();
- ❖ echo "Número de argumentos en facilit: \$numargs
";
- ❖ if (\$numargs >= 2) {
- ❖ echo "Segundo argumento es: " . func_get_arg(1) . "
";
- ❖ }
- ❖ }
- ❖ //Si arg_num>número de argumentos se generará un aviso devolverá FALSE
- ❖ **facilit(4, 5, 6);**

05Funciones.php



Funciones



//Lista de longitud variable de parámetros

```
❖ //int func_get_args(void)
❖ echo "func_get_args<br>";
❖ function facilito(){
❖     $numargs = func_num_args();
❖     echo "Número de argumentos en facilito: $numargs<br>\n";
❖     if ( $numargs >= 2 ) {
❖     echo "Segundo argumento es: " . func_get_arg( 1 ) . "<br>";
❖     }
❖     $arg_list = func_get_args();
❖     for ($i=0;$i<$numargs;$i++){
❖         echo "Argumento $i es :".$arg_list[$i]."<br>";
❖     }
❖ }
❖ facilito (7,8,9);
```

05Funciones.php



Cadenas de Caracteres



<code>chr(int valor)</code>	Devuelve el carácter asociado
<code>ord(carácter)</code>	Devuelve el número ascii asociado a un carácter
<code>strtolower(\$var)</code> y <code>strtoupper(\$var)</code>	Convierte a minúscula y mayúscula respectivamente
<code>strpos(\$var,\$subcad)</code>	Devuelve la posición de la subcadena, es false si no la encuentra
<code>split(separador,cadena)</code>	Divide una cadena en varias usando un carácter separador
<code>substr(cadena, inicio, longitud).</code>	Devuelve una subcadena de otra, empezando por inicio y de longitud longitud.
<code>chop(cadena)</code>	Elimina los saltos de línea y los espacios finales de una cadena.
<code>str_replace(cadena1, cadena2, texto)</code>	Reemplaza la cadena1 por la cadena2 en el texto.



Cadenas de Carácteres



<code>strlen(cadena)</code>	Devuelve el número de caracteres de una cadena.
<code>trim (cadena)</code>	Elimina espacios del principio y final de una cadena
<code>strstr(cadena_a_buscar,cadena_de_busqueda)</code>	Encuentra la primera aparición de una cadena
Funciones Avanzadas	
<code>ereg(cadenaPatron,cadenaFuente,[array de registros])</code>	Coincidencia de expresiones regulares : <code>if(ereg("^(.+)@(.+)\\.(.+)\$",\$email,\$arr)) {...}</code>
<code>eregi</code>	Sin diferenciar mayúsculas y minúsculas
<code>ereg_replace(cadenaPatrón,cadenaReemplazar,cadenaNueva,)</code>	Reemplaza expresiones regulares <code>\$string = ereg_replace("^","
",\$string);</code> <code>/* Coloca la etiqueta
 al comienzo de \$string. */</code>
<code>eregi_replace</code>	Sin diferenciar mayúsculas y minúsculas

- ❖ Se definen usando el constructor array
 - `$person = array ("Job", 5=> "Isaias", "Ezequiel");`

- ❖ Asigando valores a cada elemento:
 - `$MyArray[] = 'Hola';`
 - `$MyArray[] = 'Mundo Cristiano';`

- ❖ Tipos de arreglos:
 - **Arreglos enumerados o indexados numéricamente**
 - **Arreglos asociativos: índices cadenas de caracteres**

```
$ciudades = array("nin" => "Ninive", "is" => "Israel", "si"=>"Sión");
```

- ❖ **Funciones relacionadas:**
- ❖ `count(vector)` o `sizeof(vector)`: Devuelve número de elementos.
- ❖ `next`, `reset`, `prev`, `currenty end`: Se desplaza por el vector.
 - **current** -Devuelve el valor del elemento que indica el puntero
 - **pos** - realiza la misma función que **current**
 - **reset** - mueve el puntero al primer elemento de la tabla
 - **end** - mueve el puntero al último elemento de la tabla
 - **next** - mueve el puntero al elemento siguiente
 - **prev** - mueve el puntero al elemento anterior
- ❖ `array_splice(vector, pos ini, tamaño)`: Elimina la posición de un elemento.



❖ ARRAY INDEXADO

08Arrays.php

- ❖ `$person = array ("Job", "Isaias", "Ezequiel");`
- ❖ `//FOR`
- ❖ `$IMax = count($person);`
- ❖ `for ($i=0;$i<$IMax;$i++)`
- ❖ `{ echo "elemento $i : $person[$i]
";`
- ❖ `}`
- ❖ `//FOR-EACH`
- ❖ `//foreach ($person as $new){`
- ❖ `foreach ($person as $key=>$new){`
- ❖ `print "array[$key] = $new
";`
- ❖ `}`

❖ //CURRENT-NEXT

```
❖ $IMax = count($persona);  
❖ reset($persona);  
❖ for($i=0; $i<=$IMax; $i++){  
❖     echo current($persona)."<br>";  
❖     next($persona);  
❖ }
```

❖ //DO-WHILE

```
❖ $person = array ("Job", "Isaias", "Ezequiel");  
    //Se declara para colocar el puntero en posición 0  
  
❖ do {  
❖     $key   = key($person);  
❖     $value = current($person);  
❖     print("El elemento ".$key. " contiene el valor ".$value."<br>\n");  
❖ } while (next($person));
```

08Arrays.php



❖ ARRAY ASOCIATIVOS

```
❖ $visitas = array("lunes"=>200, "martes"=>186,  
❖ "miércoles"=>190, "jueves"=>175);
```

```
❖ //FOR
```

```
❖ $IMax = count($visitas );
```

```
❖ reset $visitas );
```

```
❖ for($i=0;$i<=$IMax;$i++){
```

```
❖     echo current $visitas )."<br>";
```

```
❖     next $visitas );
```

```
❖ }
```

```
❖ //FOR-EACH
```

```
❖ //foreach ($visitas as $new){
```

```
❖ foreach ($visitas as $key=>$new){
```

```
❖     print "array[$key] = $new<br>";
```

```
❖ }
```

08Arrays.php

- ❖ //DO-WHILE
- ❖ \$visitas = array("lunes"=>200, "martes"=>186,
- ❖ "miércoles"=>190, "jueves"=>175);
- ❖ //Se declara para colocar el puntero en posición 0
- ❖ do {
- ❖ \$key = key(\$visitas);
- ❖ \$value = current(\$visitas);
- ❖ print("El elemento ".\$key. " contiene el valor ".\$value."
\n");
- ❖ } while (next(\$visitas));

- ❖ //WHILE
- ❖ reset(\$visitas);
- ❖ while (list(\$clave, \$valor) = each(\$visitas))
- ❖ {
- ❖ echo "el día \$clave ha tenido \$valor visitas
";

- ❖ }

08Arrays.php



Expresiones Regulares



- ❖ Las expresiones regulares permiten la búsqueda de patrones complejos dentro de una cadena.

<code>/adorador/</code>	Se ajusta a la cadena
<code>/[aeiou]/</code>	Coincide con cualquier carácter dentro del grupo
<code>[^aeiou]</code>	Coincide con cualquier carácter que no pertenezca al grupo
<code>[0-9]</code>	Coincide con cualquier carácter dentro del grupo
<code>[0-9-]</code>	Coincidan dígitos y el guión.
<code>[\-0-9]</code>	\ describe caracteres especiales (-)



Expresiones Regulares



<code>/\d/</code>	Conjunto de caracteres alfanuméricos : [0-9]
<code>/\D/</code>	Conjunto de caracteres no-alfanuméricos: [^0-9]
<code>/\w/</code>	Conjunto de caracteres alfanuméricos : [0-9,A-Z,a-z]
<code>/\W/</code>	Conjunto de caracteres no-alfanuméricos: [^0-9A-Za-z]
<code>/\s/</code>	Conjunto de caracteres que definen espacios en blanco (el tabulador, nueva línea y el retorno de carro) : [\t\n\r]
<code>/\S/</code>	Conjunto de caracteres no-espacios en blanco: [^ \t\n\r]
<code>*</code>	Coincide con cero o más veces <code>cant*a</code> coincide con <code>canta</code> , <code>cana</code> , <code>cantttta</code>
<code>+</code>	Coincide con uno o más (al menos uno) <code>cant+a</code> coincide con <code>canta</code> , <code>canttttta</code> , NO coincide con <code>cana</code>
<code>?</code>	Coincide con cero o una vez <code>cant?a</code> coincide con <code>canta</code> y <code>cana</code>
<code>{n}</code>	Coincide con n caracteres <code>a{5}</code> exactamente 5 letras "a"



Expresiones Regulares



$\{n,\}$	Coincide con al menos n caracteres $a\{2,\}$ coincide con cualquier palabra que tenga al menos dos "a" o más: casaa o casaaaaa, no con casa
$\{n,m\}$	Coincide con al menos n y un máximo de m caracteres $a\{2,3\}$ coincide con casaa, casaaa
*	equivale a $\{0,\}$ (0 ó más veces)
+	equivale a $\{1,\}$ (1 ó más veces)
?	equivale a $\{0,1\}$ (0 ó 1 vez)
$./$	Se ajusta a cualquier carácter
$/\^/$	Simboliza al comienzo de la cadena
$/\$/$	Simboliza al final de una cadena



Expresiones Regulares



(xyz)	Coincide con la secuencia exacta xyz
x y	Coincide si esta presente x ó y
Modificador i	Trata igual a mayúscula y a minúscula /Sonrie/i : Se ajustará a sonrie , o a SoNrIe
Modificador g	No para cuando encuentra una coincidencia sino que sigue evaluando el resto de la cadena. Es útil cuando no queremos buscar sino reemplazar /grandeza/g

- ❖ `/^\d+$/`
- ❖ `/^[A-Z,a-z,0-9,_]+\.{0,1}[A-Z,a-z,_]+\@[A-Z,a-z,_]+\.[A-Z,a-z,_]+/`
- ❖ `/^http[s]?://\w[\.\w]+$/`
- ❖ `/^\w-\.[\w-]+\.[\w-]{2,4}$/`

No permite dominios de menos de 2 caracteres ni de más de 4.



PHP

- ❖ `<?php`
- ❖ `$email = "webmaster@anaya.es";`
- ❖ `if(ereg("^(.+)+@(.+)\.(.+)$", $email, $arr)) {`
- ❖ `print("La dirección de email es correcta.
");`
- ❖ `print("Dirección: ".$arr[0]."
");`
- ❖ `print("Usuario: ".$arr[1]."
");`
- ❖ `print("Servidor: ".$arr[2]."
");`
- ❖ `print("Dominio: ".$arr[3]."
");`
- ❖ `} else {`
- ❖ `printf("La dirección es incorrecta");`
- ❖ `}`
- ❖ `?>`



PHP

- ❖ `<?php`
- ❖ `$date = "2006-12-24";`
- ❖ `if (ereg("[0-9]{4}-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {`
- ❖ `echo "$regs[3].$regs[2].$regs[1]";`
- ❖ `}`
- ❖ `else {`
- ❖ `echo " Formato de Fecha Inválida: $date";`
- ❖ `}`
- ❖ `?>`



JavaScript

- ❖ Objeto RegExp: Expresión Regular.

```
var reg = / ^\d+$/
```

```
var reg = new RegExp ("^\d+$ ")
```

- ❖ Métodos de las Expresiones Regulares:

test	Un método de RegExp que prueba si existe una coincidencia en una cadena. Devuelve verdadero o falso.
split(regex)	Un método de String que emplea una expresión regular para separarla por los delimitadores indicados en la expresión regular.
search(regex)	Un método de String que prueba si existe una coincidencia en una cadena. Devuelve la posición de la coincidencia, o -1 si la búsqueda falla.
replace(regex,r eemplazo)	Un método de String que realiza una búsqueda de una coincidencia en una cadena, y reemplaza la subcadena coincidente con la subcadena de reemplazo.



JavaScript

```
❖ function isNumerico(num,bEmptyOK){  
❖   var reNumeric = /^\\d+$/;  
  
❖     if (trim(num)!=" " && !reNumeric.test(num)){  
❖       return true;  
❖     else  
❖       return false;  
❖   }  
❖   return true;  
❖ }
```