UT 2: Desarrollo de aplicaciones web con PHP

2ºDAW – Desarrollo Entorno Servidor

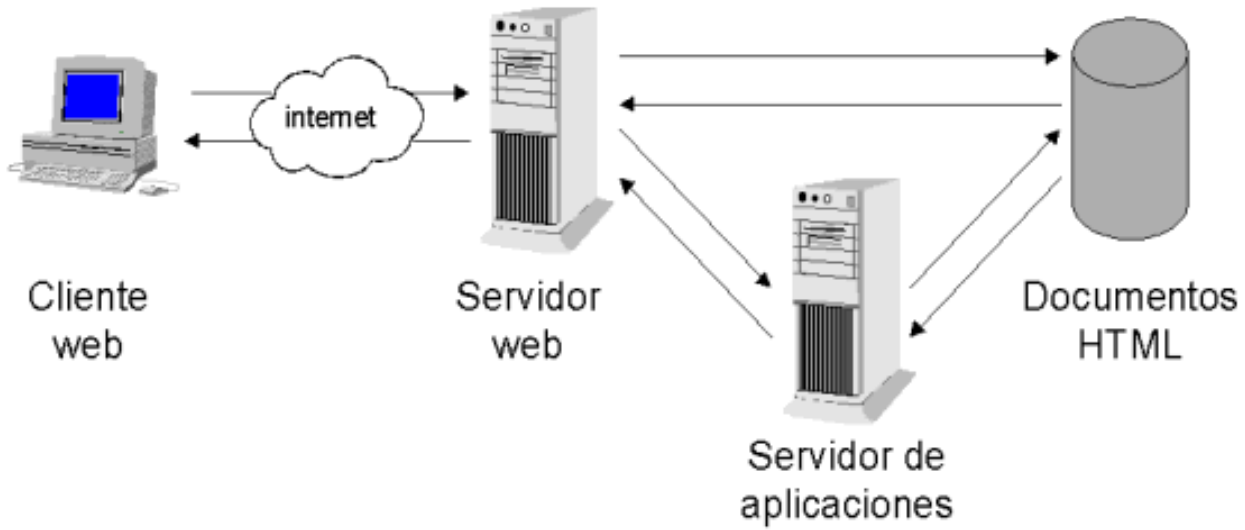
Alfonso Rebolleda Sánchez

Lenguaje PHP

PHP es un lenguaje de script del lado del servidor.

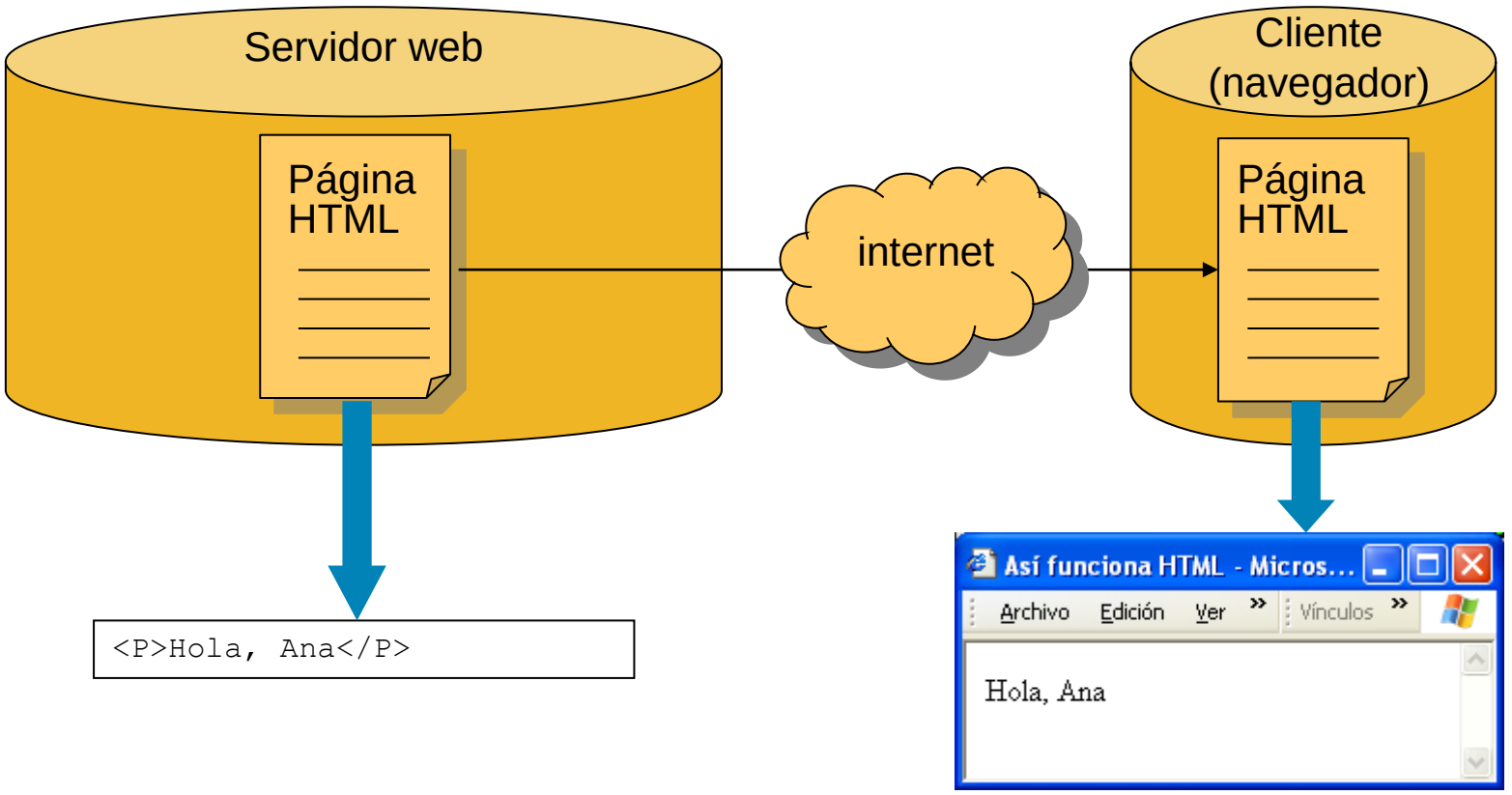
Los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta y ejecuta antes de servir las páginas al cliente.

El cliente no ve el código PHP sino los resultados que produce.



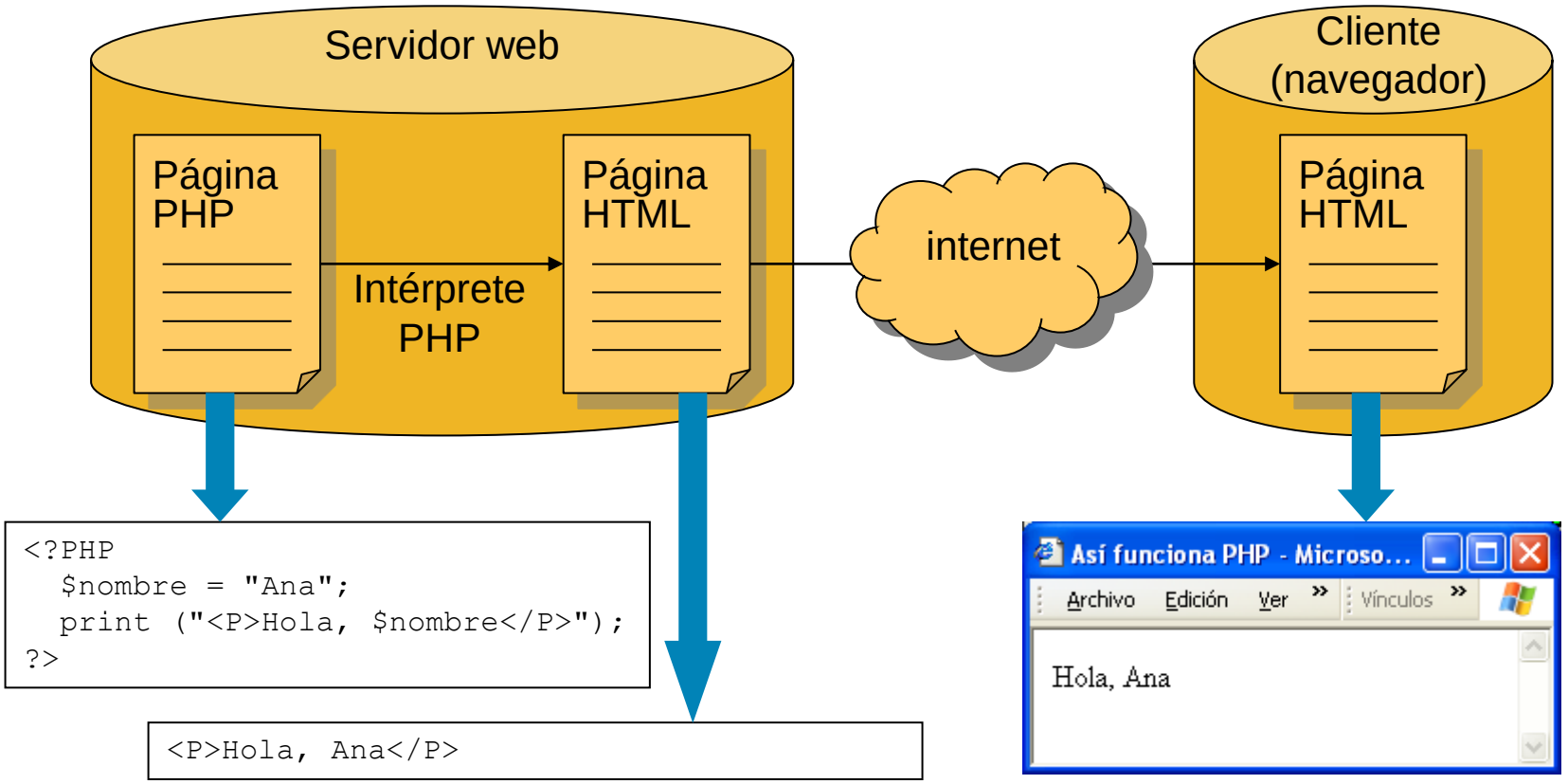
Lenguaje PHP

Funcionamiento PHP



Lenguaje PHP

Funcionamiento PHP



Lenguaje PHP

1. Sintaxis básica
2. Tipos de datos
3. Variables
4. Constantes
5. Expresiones y operadores
6. Estructuras de control
7. Funciones
8. Tablas
9. Bibliotecas de funciones

Sintaxis básica

- PHP es sensible a las mayúsculas
- ¿Cómo se incrusta en la página web?

  `<?PHP ... ?>`

  recomendado, siempre disponible

  `<?= expresión ?>`

  equivale a <? echo expresión ?>
- Las instrucciones se separan con un ; como en C. La marca final ?> implica un ;
- Comentarios: como en C, /* … */ (varias líneas) y // (una línea)

```
/* Comentario de
varias líneas */
print “hola”; // Comentario de una línea
```

Sintaxis básica

- Para imprimir: **echo** y **print**

  echo: muestra una o más cadenas
  echo cadena1 [, cadena2…];

```
echo “Hola mundo”;
echo “Hola “, “mundo”;
```

  print: muestra una cadena
  print cadena;

```
print “Hola mundo”;
print  “Hola “ . “mundo”;
```

Sintaxis básica

- Ejemplo:

```
<HTML>
<HEAD>
<TITLE>Mi primer programa en PHP</TITLE>
</HEAD>

<BODY>

<?PHP
   print (“<P>Hola mundo</P>”);
?>

</BODY>
</HTML>
```

Sintaxis básica

- Uso de \n para generar código HTML legible
- a) Sin \n

Código PHP

```
print ("<P>Párrafo 1</P>");
print ("<P>Párrafo 2</P>");
```

Código HTML

```
<P>Párrafo 1</P><P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica

- Uso de \n para generar código HTML legible
- b) Con \n

Código PHP

```
print ("<P>Párrafo 1</P>\n");
print ("<P>Párrafo 2</P>\n");
```

Código HTML

```
<P>Párrafo 1</P>
<P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica

- Inclusión de ficheros externos:
  - **include**()
  - **require**()
- Ambos incluyen y evalúan el fichero especificado
- Diferencia: en caso de error include() produce un warning y require() un error fatal
- Se usará require() si al producirse un error debe interrumpirse la carga de la página

Sintaxis básica

Ejemplo:

```
<HTML>
<HEAD>
   <TITLE>Título</TITLE>
<?PHP
// Incluir bibliotecas de funciones
   require ("conecta.php");
   require ("fecha.php");
   require ("cadena.php");
   require ("globals.php");
?>
</HEAD>
<BODY>
<?PHP
   include ("cabecera.html");
?>
// Código HTML + PHP
. . .
<?PHP
   include ("pie.html");
?>
</BODY>
</HTML>
```

Tipos Datos

- PHP soporta 8 **tipos de datos primitivos**:
  - Tipos escalares: boolean, integer, double, string
  - Tipos compuestos: array, object
  - Tipos especiales: resource, NULL
- El tipo de una variable no se suele especificar. Se decide en tiempo de ejecución en función del contexto y puede variar
- Funciones de interés:
  - La función gettype() devuelve el tipo de una variable
  - Las funciones is_*type* comprueban si una variable es de un tipo dado:

    is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar(),

    is_string()
  - La función var_dump() muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays

Tipos Datos

- Tipo **integer** (números enteros)
  - 27, -5, 0
- Tipo **double** (números reales)
  - 1.234, -5.33
- Tipo **boolean** (lógico)
  - Valores: *true*, *false* (insensibles a las mayúsculas)
  - El 0 y la cadena vacía tienen valor *false*

Tipos Datos

- Tipo string:
  - Las cadenas se encierran entre comillas simples o dobles:
    - ‘simples’: admite los caracteres de escape \’ (comilla simple) y \\ (barra). Las variables **NO** se expanden
    - “dobles”: admite más caracteres de escape, como \n, \r, \t, \\, \$, \”. Los nombres de variables **SÍ** se expanden
    - Ejemplos:

```
$a = 9;
print ‘a vale $a\n’;
   // muestra a vale $a\n
print “a vale $a\n”;
   // muestra a vale 9 y avanza una línea
print “<IMG SRC=‘logo.gif’>”;
   // muestra <IMG SRC=‘logo.gif’>
print “<IMG SRC=\”logo.gif\”>”;
   // muestra <IMG SRC=“logo.gif”>
```

  - Acceso a un carácter de la cadena:
    - La forma es $inicial = $nombre{0};

Variables

- Las variables siempre van precedidas de un $
- El nombre es sensible a las mayúsculas
- Comienzan por letra o subrayado, seguido de letras, números o subrayado
- Variables predefinidas:

  $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIES, $_FILES, $_ENV, $_REQUEST, $_SESSION
- Ámbito: globales al fichero (excepto funciones) o locales a una función
- Ejemplo:

```
$valor = 5;
print “El valor es: “ . $valor . “\n”;
print “El valor es: $valor\n”; // ojo: comillas dobles

Resultado:
   El valor es: 5
```

Variables

- Variables variables
  - Se pueden crear nombres de variables dinámicamente
  - La variable variable toma su nombre del valor de otra variable previamente declarada
  - Ejemplo:

```
$a = "hola";
$$a = "mundo";

print "$a $hola\n";
print "$a ${$a}";

Resultado:
     hola mundo
     hola mundo
```

Variables

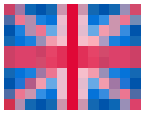
- Ejemplo de variables variables: página internacionalizada (1)

```
<?PHP
   $mensaje_es="Hola";
   $mensaje_en="Hello";
   $idioma = "es";
   $mensaje = "mensaje_" . $idioma;
   print $$mensaje;
?>
```

Variables

- Ejemplo de variables variables: página internacionalizada (2)

```
<?PHP
   $mensaje_es="Hola";
   $mensaje_en="Hello";
   $idioma = "en";
   $mensaje = "mensaje_" . $idioma;
   print $$mensaje;
?>
```



Constantes

- Definición de constantes:

```
define (“CONSTANTE”, “hola”);
print CONSTANTE;
```

- No llevan $ delante
- Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)

Expresiones y Operadores

- Operadores aritméticos:
  +, -, *, /, %, ++, --
- Operador de asignación:
  =
  operadores combinados: .=, +=, etc
  $a = 3; $a += 5; → a vale 8
  $b = “hola ”; $b .= “mundo”; → b vale “hola mundo”
  → Equivale a $b = $b . “mundo”;
- Operadores de comparación:
  ==, !=, <, >, <=, >= y otros
- Operador de control de error: @. Antepuesto a una expresión, evita cualquier mensaje de error que pueda ser generado por la expresión
- Operadores lógicos:
  and (&&), or (||), !, xor
  and/&& y or/|| tienen diferentes prioridades
- Operadores de cadena:
  concatenación: . (punto)
  asignación con concatenación: .=

Expresiones y Operadores

- Precedencia de operadores (de mayor a menor):

```
++, --
*, /, %
+,-
<, <=, >, >=
==, !=
&&
||
and
or
```

Estructuras de Control

- Estructuras selectivas:
  - if-else
  - switch
- Estructuras repetitivas:
  - while
  - for
  - foreach

Estructuras de Control

- Estructura selectiva **if-else**

```
if (condición)
   sentencia
```

```
if (condición)
   sentencia 1
else
   sentencia 2
```

```
if (condición1)
   sentencia 1
else if (condición2)
   sentencia 2
...
else if (condición n)
   sentencia n
else
   sentencia n+1
```

- Mismo comportamiento que en C
- Las sentencias compuestas se encierran entre llaves
- elseif puede ir todo junto

Estructuras de Control

- Ejemplo de estructura selectiva if-else:

```
<?PHP
   if ($sexo == 'M')
        $saludo = "Bienvenida, ";
   else
        $saludo = "Bienvenido, ";
   $saludo = $saludo . $nombre;
   print ($saludo);
?>
```

Estructuras de Control

- Estructura selectiva **switch**

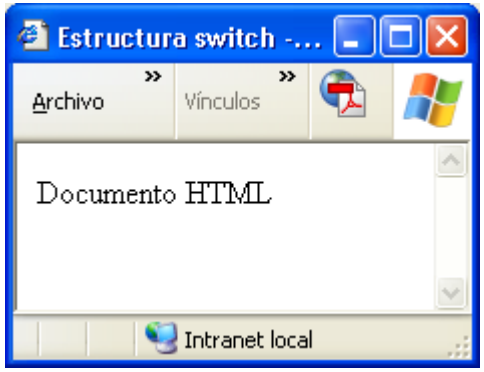
```
switch (expresión)
{
   case valor_1:
        sentencia 1
        break;
   case valor_2:
        sentencia 2
        break;
   …
   case valor_n:
        sentencia n
         break;
   default
        sentencia n+1
}
```

- Mismo comportamiento que en C, sólo que la expresión del case puede ser integer, float o string

Estructuras de Control

- Ejemplo de estructura selectiva switch:

```
switch ($extension)
{
   case ("PDF"):
      $tipo = "Documento Adobe PDF";
      break;
   case ("TXT"):
      $tipo = "Documento de texto";
      break;
   case ("HTML"):
   case ("HTM"):
      $tipo = "Documento HTML";
      break;
   default:
      $tipo = "Archivo " . $extension;
}
print ($tipo);
```

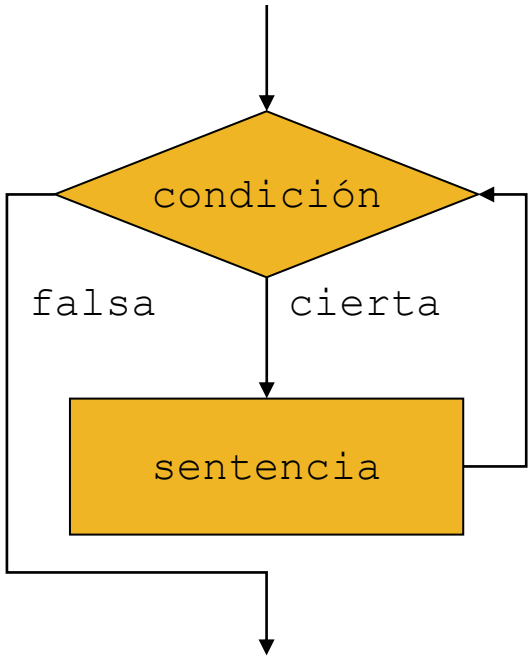


Estructuras de Control

- Estructura repetitiva **while**

```
while (condición)
   sentencia
```

- Mismo comportamiento que en C



Estructuras de Control

- Ejemplo de estructura repetitiva while:

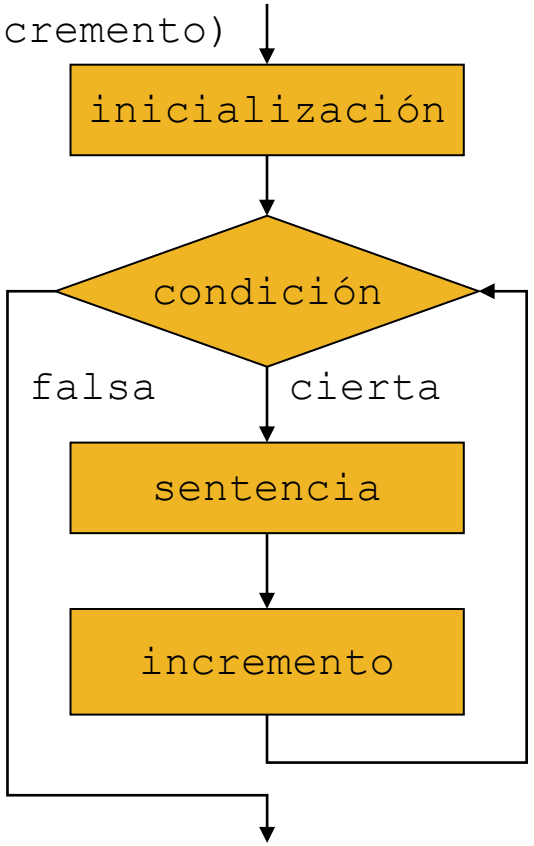
```
<?PHP
   print ("<UL>\n");
   $i=1;
   while ($i <= 5)
   {
        print ("<LI>Elemento $i</LI>\n");
        $i++;
   }
   print ("</UL>\n");
?>
```

Estructura de Control

- Estructura repetitiva **for**

```
for (inicialización; condición; incremento)
   sentencia
```

- Mismo comportamiento que en C



Estructura de Control

- Ejemplo de estructura repetitiva for:

```
<?PHP
   print ("<UL>\n");
   for ($i=1; $i<=5; $i++)
        print ("<LI>Elemento $i</LI>\n");
   print ("</UL>\n");
?>
```

Funciones

- Ejemplo:

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

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

Funciones

- Por defecto los parámetros se pasan por valor
- Paso por referencia:

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

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

Funciones

- Argumentos por defecto

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

- Salida:

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

Funciones

- 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:
```

Tablas

- Sintaxis:

```
array ([clave =>] valor, ...)
```

- La clave es una cadena o un entero no negativo. El valor puede ser de cualquier tipo válido en PHP, incluyendo otro array
- Ejemplos:

```
$color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);
$medidas = array (10, 25, 15);
```

- Acceso:

```
$color['rojo'] // No olvidar las comillas
$medidas[0]
```

- El primer elemento es el 0

Tablas

- La estructura de control **foreach** permite iterar sobre arrays
- Sintaxis:

```
foreach (expresión_array as $valor)
   sentencia
foreach (expresión_array as $clave => $valor)
   sentencia
```

- Ejemplos:

```
foreach ($color as $valor)
   print "Valor: $valor<BR>\n";
foreach ($color as $clave => $valor)
   print "Clave: $clave; Valor: $valor<BR>\n";
```

- Salida:

```
Valor: 101
Valor: 51
Valor: 255
Clave: rojo; Valor: 101
Clave: verde; Valor: 51
Clave: azul; Valor: 255
```

Bibliotecas 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

Bibliotecas Funciones

- Funciones de manipulación de cadenas
  - explode()
    - Divide una cadena en subcadenas
    - array **explode** (string separator, string string [, int limit])
  - rtrim(), ltrim(), trim()
    - Eliminan caracteres a la derecha, a la izquierda o por ambos lados de una cadena
    - string **rtrim** (string str [, string charlist])
  - strstr()
    - Busca la primera ocurrencia de una subcadena
  - strtolower() / strtoupper()
    - Convierte una cadena a minúscula / mayúscula
  - strcmp() / strcasecmp()
    - Compara dos cadenas con/sin distinción de mayúsculas
  - strlen()
    - Calcula la longitud de una cadena

Bibliotecas Funciones

- Funciones de fecha y hora
  - date()
    - Formatea una fecha según un formato dado
    - Ejemplo:

```
$fecha  = date ("j/n/Y H:i");
print ("$fecha");
Resultado:
     26/9/2005 17:36
```

  - strtotime()
    - Convierte una fecha en un *timestamp* de UNIX
    - Ejemplo:

```
$fecha = date ("j/n/Y", strtotime("5 april 2001"));
print ("$fecha");
Resultado:
     5/4/2001
```

Bibliotecas Funciones

- Funciones de arrays
  - array_count_values()
    - Calcula la frecuencia de cada uno de los elementos de un array
  - array_search()
    - Busca un elemento en un array
  - count()
    - Cuenta los elementos de un array
  - sort(), rsort()
    - Ordena y reindexa un array (r=decreciente)
  - ksort(), krsort()
    - Ordena por claves un array (r=decreciente)

Formularios

1. Acceso a formularios HTML desde PHP
2. El formulario de PHP
3. Subida de ficheros al servidor
4. Validación de los datos de un formulario

Formularios – Acceso desde PHP

- Desde PHP se puede acceder fácilmente a los datos introducidos desde un formulario HTML
- Veámoslo con un ejemplo simple

Formularios – Acceso desde PHP

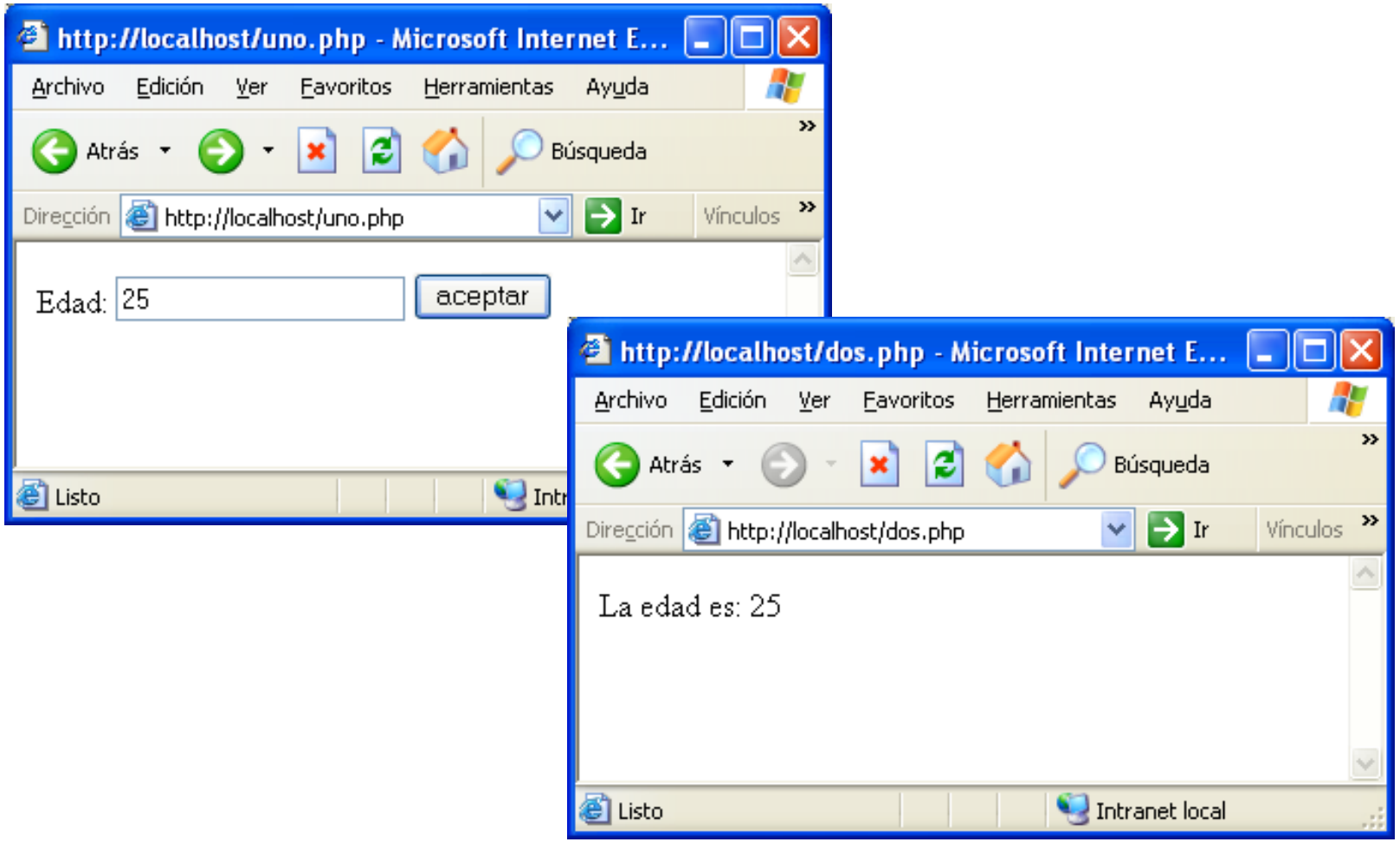
- Fichero uno.php

```
<HTML>
<BODY>
<FORM ACTION=”dos.php” METHOD=”POST”>
   Edad: <INPUT TYPE=”text” NAME=”edad”>
   <INPUT TYPE=”submit” VALUE=”aceptar”>
</FORM>
</BODY>
</HTML>
```

- Fichero dos.php

```
<HTML>
<BODY>
<?PHP
   print (“La edad es: $edad”);
?>
</BODY>
</HTML>
```

Formularios – Acceso desde PHP



Formularios – Acceso desde PHP

- A partir de PHP 4.2.0, el valor por defecto de la directiva de PHP **register_globals** es off
- Esto tiene una gran importancia sobre los formularios, ya que no es posible acceder a las variables enviadas de la manera anterior (como variables globales). En su lugar hay que utilizar la variable predefinida de PHP **$_REQUEST**, escribiendo $_REQUEST[‘edad’] en lugar de $edad
- Se puede poner register_globals = on en el fichero de configuración php.ini, pero no es recomendable por motivos de seguridad. Una alternativa que permite hacer mínimos cambios en el código ya existente es la siguiente:

```
$edad = $_REQUEST[‘edad’];
```

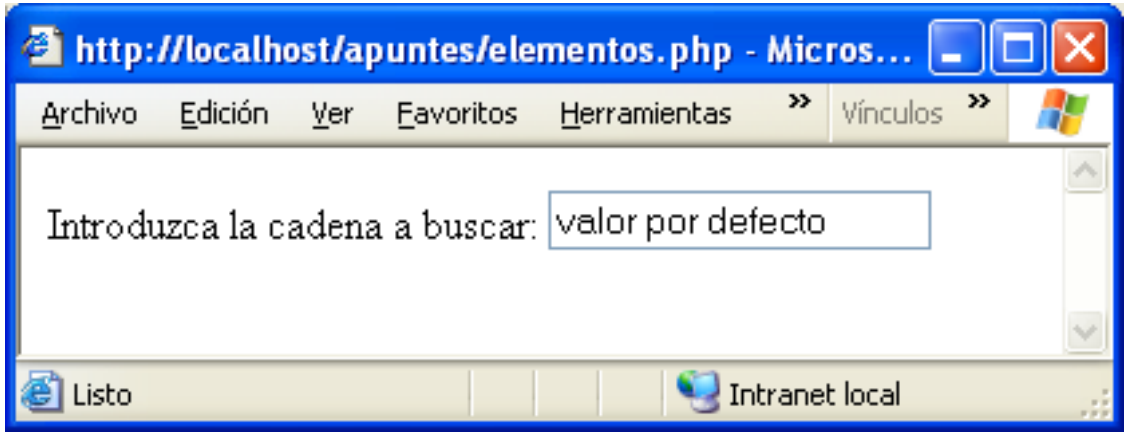
Formularios – Acceso desde PHP

- Acceso a los diferentes tipos de elementos de entrada de formulario
  - Elementos de tipo INPUT
    - TEXT
    - RADIO
    - CHECKBOX
    - BUTTON
    - FILE
    - HIDDEN
    - PASSWORD
    - SUBMIT
  - Elemento SELECT
    - Simple / múltiple
  - Elemento TEXTAREA

Formularios – Acceso desde PHP

- TEXT

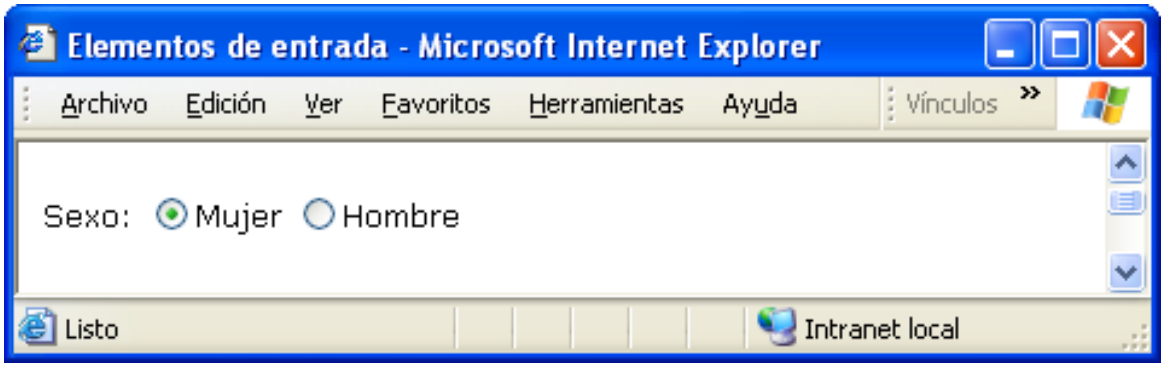
```
Introduzca la cadena a buscar:
<INPUT TYPE="text" NAME="cadena" VALUE="valor por defecto" SIZE="20">


<?PHP
   $cadena = $_REQUEST['cadena'];
   print ($cadena);
?>
```



Formularios – Acceso desde PHP

- RADIO

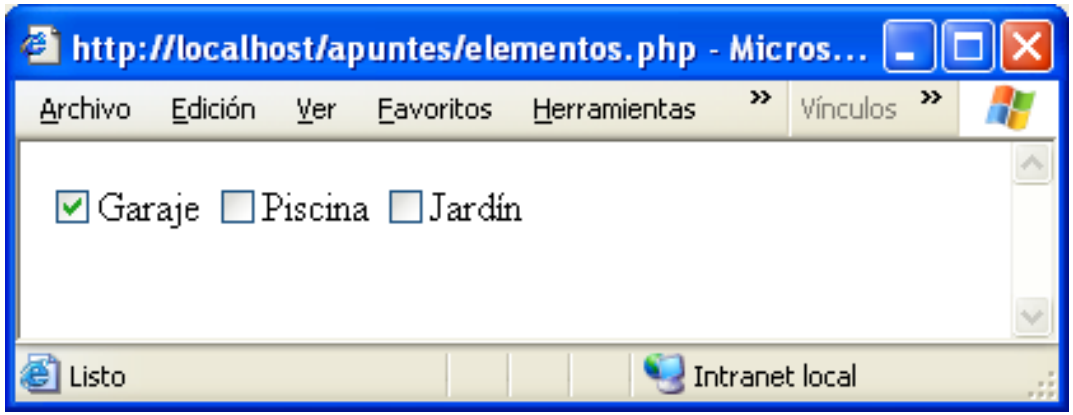
```
Sexo:
<INPUT TYPE="radio" NAME="sexo" VALUE="M" CHECKED>Mujer
<INPUT TYPE="radio" NAME="sexo" VALUE="H">Hombre


<?PHP
   $sexo = $_REQUEST['sexo'];
   print ($sexo);
?>
```



Formularios – Acceso desde PHP

- CHECKBOX

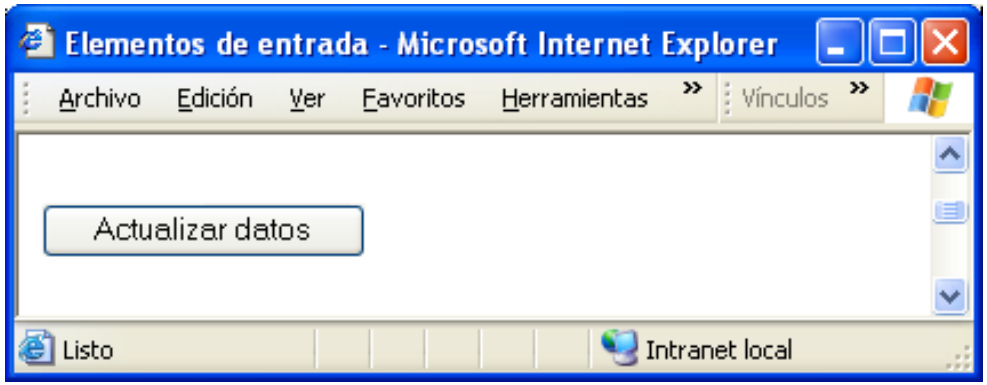
```
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="garaje" CHECKED>Garaje
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="piscina">Piscina
<INPUT TYPE="checkbox" NAME="extras[]" VALUE="jardin">Jardín


<?PHP
   $extras = $_REQUEST['extras'];

   foreach ($extras as $extra)
      print ("$extra<BR>\n");
 ?>
```



Formularios – Acceso desde PHP

- BUTTON

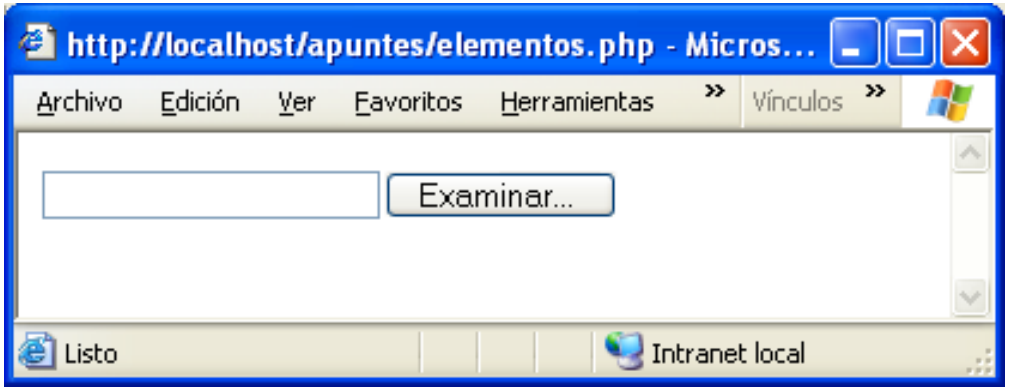
```
<INPUT TYPE="button" NAME="actualizar" VALUE="Actualizar datos">


<?PHP
   $actualizar = $_REQUEST['actualizar'];
   if ($actualizar)
      print ("Se han actualizado los datos");
?>
```



Formularios – Acceso desde PHP

- FILE

```
<FORM ACTION="procesa.php" METHOD="post"
   ENCTYPE="multipart/form-data">
   <INPUT TYPE="file" NAME="fichero">
</FORM>
```

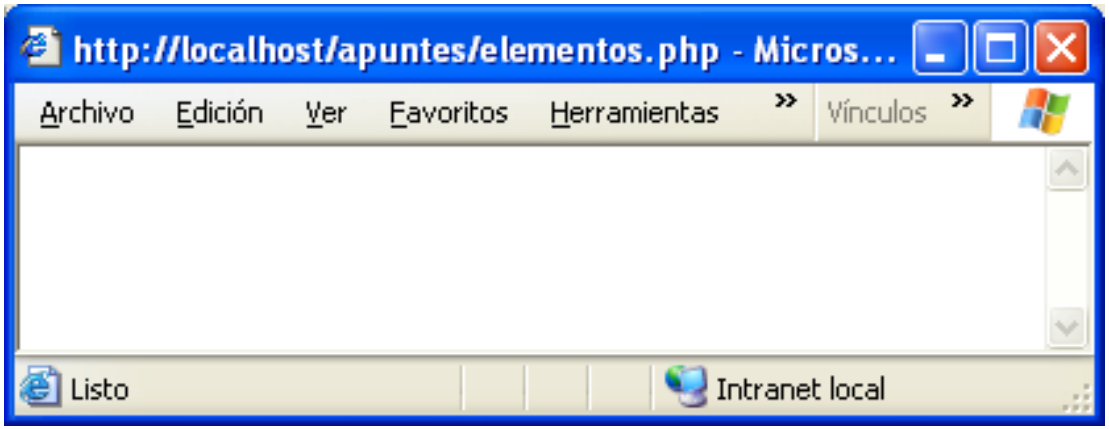


Formularios – Acceso desde PHP

- HIDDEN

```
<?PHP
   print("<INPUT TYPE='hidden' NAME='username' VALUE='$usuario'>\n");
?>
```

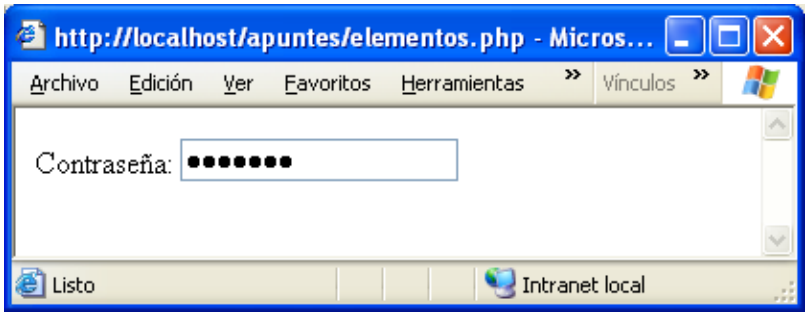
```
<?PHP
   $username = $_REQUEST['username'];
   print ($username);
?>
```



Formularios – Acceso desde PHP

- PASSWORD

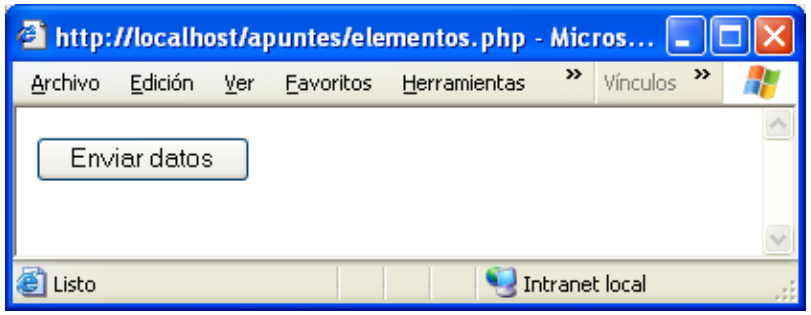
```
Contraseña: <INPUT TYPE="password" NAME="clave">


<?PHP
   $clave = $_REQUEST['clave'];
   print ($clave);
?>
```



Formularios – Acceso desde PHP

- SUBMIT

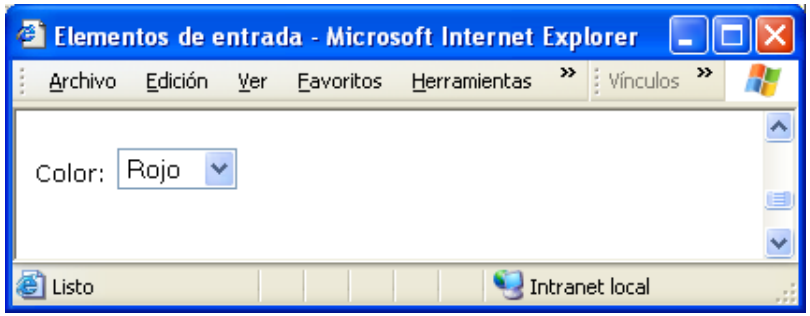
```
<INPUT TYPE="submit" NAME="enviar" VALUE="Enviar datos">


<?PHP
   $enviar = $_REQUEST['enviar'];
   if ($enviar)
      print ("Se ha pulsado el botón de enviar");
?>
```



Formularios – Acceso desde PHP

- SELECT simple

```
Color:
<SELECT NAME="color">
   <OPTION VALUE="rojo" SELECTED>Rojo
   <OPTION VALUE="verde">Verde
   <OPTION VALUE="azul">Azul
</SELECT>


<?PHP
   $color = $_REQUEST['color'];
   print ($color);
?>
```



Formularios – Acceso desde PHP

- SELECT múltiple

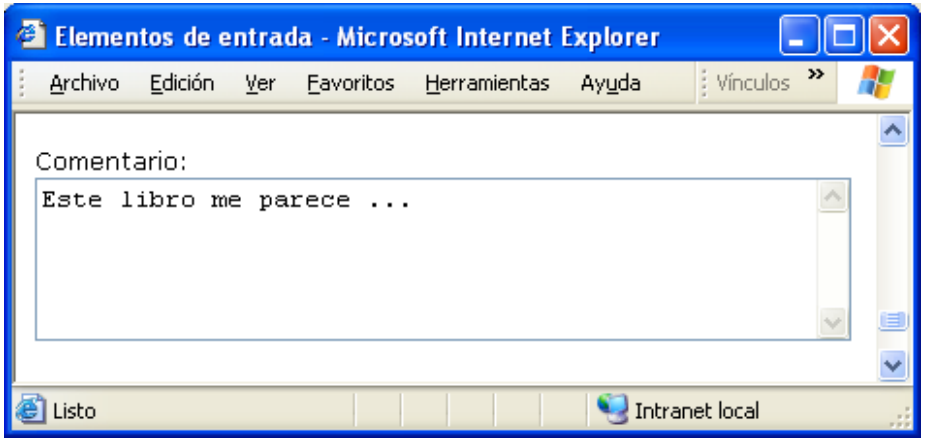
```
Idiomas:
<SELECT MULTIPLE SIZE="3" NAME="idiomas[]">
   <OPTION VALUE="ingles" SELECTED>Inglés
   <OPTION VALUE="frances">Francés
   <OPTION VALUE="aleman">Alemán
   <OPTION VALUE="holandes">Holandés
</SELECT>
```

```
<?PHP
   $idiomas = $_REQUEST['idiomas'];
   foreach ($idiomas as $idioma)
      print ("$idioma<BR>\n");
?>
```

Formularios – Acceso desde PHP

- TEXTAREA

```
Comentario:
<TEXTAREA COLS="50" ROWS="4" NAME="comentario">
Este libro me parece ...
</TEXTAREA>


<?PHP
   $comentario = $_REQUEST['comentario'];
   print ($comentario);
?>
```



Formularios – Gestión

- La forma habitual de trabajar con formularios en PHP es utilizar un único programa que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
- Ventajas:
  - Disminuye el número de ficheros
  - Permite validar los datos del formulario en el propio formulario
- Procedimiento:

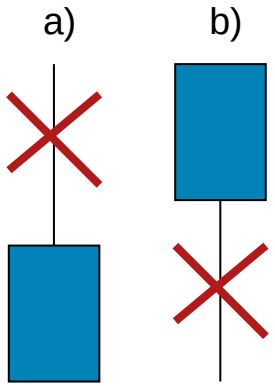
```
si se ha enviado el formulario:
   Procesar formulario
si no:
   Mostrar formulario
fsi
```

Formularios – Gestión

- Esquema de funcionamiento:

```
si se ha enviado el formulario:
   Procesar formulario
```

```
si no:
   Mostrar formulario
fsi
```



  - La 1ª vez que se carga la página se muestra el formulario (a)
  - La 2ª vez se procesa el formulario (b)

Formularios – Gestión

- Para saber si se ha enviado el formulario se acude a la variable correspondiente al botón de envío. Si este botón aparece de la siguiente forma en el formulario HTML:

```
<INPUT TYPE="SUBMIT" NAME="enviar‟ VALUE="procesar">
```

entonces la condición anterior se transforma en:

```
if (isset($enviar))
```

o bien

```
if ($enviar == “procesar”)
```

Formularios – Subidas Ficheros al Servidor

- Para subir un fichero al servidor se utiliza el elemento de entrada FILE
- Hay que tener en cuenta una serie de consideraciones importantes:
  - El elemento FORM debe tener el atributo ENCTYPE="multipart/form-data“
  - El fichero tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes:
    - En el fichero de configuración php.ini
    - En el propio formulario

Formularios – Subidas Ficheros al Servidor

php.ini

```
;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;
; Whether to allow HTTP file uploads.
file_uploads = On

; Temporary directory for HTTP uploaded files (will use
; system default if not specified).
;upload_tmp_dir =
```

```
; Maximum allowed size for uploaded files.
upload_max_filesize = 2M
```

formulario

```
<INPUT TYPE=”HIDDEN” NAME=”MAX_FILE_SIZE” VALUE='102400'>
<INPUT TYPE=”FILE” NAME="fichero">
```

Formularios – Subidas Ficheros al Servidor

- Consideraciones
  - Debe darse al fichero un nombre que evite coincidencias con ficheros ya subidos. Por ello, y como norma general, debe **descartarse el nombre original** del fichero y crear uno nuevo que sea único
  - El fichero subido se almacena en un directorio temporal y hemos de moverlo al directorio de destino usando la función move_upload_file()
- Procedimiento:

```
si se ha subido correctamente el fichero:
   Asignar un nombre al fichero
   Mover el fichero a su ubicación definitiva
si no:
   Mostrar un mensaje de error
fsi
```

Formularios – Subidas Ficheros al Servidor

HTML

```
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="102400">
<INPUT TYPE="FILE" SIZE="44" NAME="imagen">
```

- La variable $_FILES contiene toda la información del fichero subido:
  - $_FILES['imagen']['name']
    - Nombre original del fichero en la máquina cliente
  - $_FILES['imagen']['type']
    - Tipo mime del fichero. Por ejemplo, "image/gif"
  - $_FILES['imagen']['size']
    - Tamaño en bytes del fichero subido
  - $_FILES['imagen']['tmp_name']
    - Nombre del fichero temporal en el que se almacena el fichero subido en el servidor
  - $_FILES['imagen']['error']
    - Código de error asociado al fichero subido

Formularios – Subidas Ficheros al Servidor

PHP

```
if (is_uploaded_file ($_FILES['imagen']['tmp_name']))
{
   $nombreDirectorio = "img/";
   $idUnico = time();
   $nombreFichero = $idUnico . "-" . $_FILES['imagen']['name'];

   move_uploaded_file ($_FILES['imagen']['tmp_name'],
      $nombreDirectorio . $nombreFichero);
}
else
   print ("No se ha podido subir el fichero\n");
```

Formularios – Subidas Ficheros al Servidor

PHP

```php
if (is_uploaded_file ($_FILES['imagen']['tmp_name']))
{
   $nombreDirectorio = "img/";
   $nombreFichero = $_FILES['imagen']['name'];

   $nombreCompleto = $nombreDirectorio . $nombreFichero;
   if (is_file($nombreCompleto))
   {
      $idUnico = time();
      $nombreFichero = $idUnico . "-" . $nombreFichero;
   }

   move_uploaded_file ($_FILES['imagen']['tmp_name'],
      $nombreDirectorio . $nombreFichero);
}
else
   print ("No se ha podido subir el fichero\n");
```

Formularios – Validación

- Toda la información proveniente de un formulario debe considerarse por norma como contaminada, y hay que validarla antes de darla por buena y procesarla
- Lo más eficiente es mostrar los errores sobre el propio formulario para facilitar su corrección. Procedimiento:

```
si se ha enviado el formulario:
   si hay errores:
      Mostrar formulario con errores
   si no:
      Procesar formulario
   fsi
si no:
   Mostrar formulario
fsi
```

Formularios – Validación

- Este procedimiento se puede resumir para que sólo haya que mostrar una vez el formulario, bien con los valores por defecto o con los valores introducidos, y con los errores en su caso:

```
si se ha enviado el formulario:
   validar datos
fsi
si se ha enviado el formulario y no hay errores:
   Procesar formulario
si no:
   Mostrar formulario con valores por defecto o ya
   enviados
fsi
```

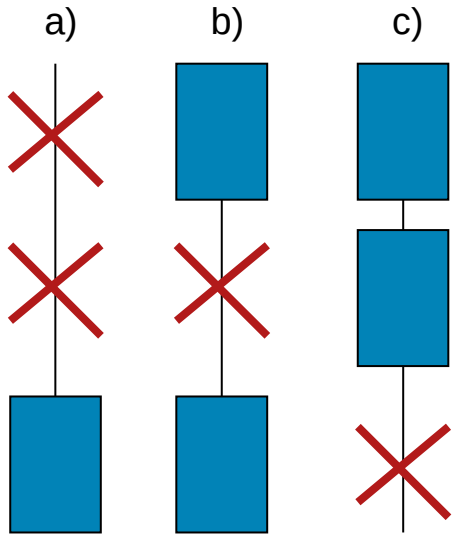
Formularios – Validación

- Esquema de funcionamiento:

```
si se ha enviado el formulario:
   validar datos
fsi
```

```
si se ha enviado y no hay errores:
   Procesar formulario
```

```
si no:
   Mostrar formulario
fsi
```



  - La 1ª vez que se carga la página se muestra el formulario (a)
  - La 2ª y sucesivas veces se validan los datos
    - Si hay errores, se muestra de nuevo el formulario con los errores (b)
    - Si no hay, se procesa el formulario (c)