Ambientes no propietarios

Introducción a PHP

Edwin Salvador

21 de abril de 2015

Sesión 3

Contenido I

Presentaciones

Temas de presentaciones?

Contenido I

Revisión de deberes

- Archivos CSS separados.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.
- Direcciones relativas no absolutas

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.
- Direcciones relativas no absolutas
- Organización de archivos por carpetas (img, js, css)

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.
- Direcciones relativas no absolutas
- Organización de archivos por carpetas (img, js, css)
- No CSS mezclado con HTML.

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.
- Direcciones relativas no absolutas
- Organización de archivos por carpetas (img, js, css)
- No CSS mezclado con HTML.
- Archivo index.php o .html

Revisión de deberes

- Archivos CSS separados.
- Archivos JS separados.
- Funcionalidad JS interesante (pocos).
- Mejorar estilo de código HTML, JS.
- Elementos obsoletos (marquee) utilizar CSS o JS mejor.
- Tamaño de imágenes y videos (youtube mejor opción).
- Mejor utilizar `<div>` en lugar de `<p>` o `<table>`.
- Identación de código.
- Direcciones relativas no absolutas
- Organización de archivos por carpetas (img, js, css)
- No CSS mezclado con HTML.
- Archivo index.php o .html
- Demasiado sencilla!!!!

Revisión de deberes

- Estructura de proyectos

Revisión de deberes

- Estructura de proyectos
  - deberes

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1
    - practica2

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1
    - practica2
    - ...

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1
    - practica2
    - ...
  - proyecto

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1
    - practica2
    - ...
  - proyecto
- No tildes, no espacio, todo minúsculas.

Revisión de deberes

- Estructura de proyectos
  - deberes
    - deber1
    - deber2
    - ...
  - practicas
    - practica1
    - practica2
    - ...
  - proyecto
- No tildes, no espacio, todo minúsculas.
- Incluir siempre un index.php o .html

Contenido I

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?
5. Si se encontrara un error en una de las herramientas open source, ¿cómo lo solucionarías?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?
5. Si se encontrara un error en una de las herramientas open source, ¿cómo lo solucionarías?
6. ¿Cuál es la diferencia entre WAMP, MAMP, LAMP?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?
5. Si se encontrara un error en una de las herramientas open source, ¿cómo lo solucionarías?
6. ¿Cuál es la diferencia entre WAMP, MAMP, LAMP?
7. ¿Qué tienen en común las direcciones `http://localhost` y `127.0.0.1`?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?
5. Si se encontrara un error en una de las herramientas open source, ¿cómo lo solucionarías?
6. ¿Cuál es la diferencia entre WAMP, MAMP, LAMP?
7. ¿Qué tienen en común las direcciones `http://localhost` y `127.0.0.1`?
8. ¿Para que sirve un programa FTP?

Preguntas

1. ¿Cuáles son los 4 elementos fundamentales para crear una página web dinámica?
2. ¿Qué significa HTML?
3. PHP y JavaScript son dos lenguajes que generan páginas dinámicas. ¿Cuál es la principal diferencia entre estos dos y porque se los usaría conjuntamente?
4. ¿Qué significa CSS y para que sirve?
5. Si se encontrara un error en una de las herramientas open source, ¿cómo lo solucionarías?
6. ¿Cuál es la diferencia entre WAMP, MAMP, LAMP?
7. ¿Qué tienen en común las direcciones `http://localhost` y `127.0.0.1`?
8. ¿Para que sirve un programa FTP?
9. ¿Cuál es la principal desventaja de trabajar directamente en un servidor web remoto?

Contenido I

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`
- El servidor automáticamente pasa los archivos `.php` al interprete de PHP.

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`
- El servidor automáticamente pasa los archivos `.php` al interprete de PHP.
- Se puede configurar el servidor de tal manera que haga lo mismo con archivos `.html` pero eso no es necesario.

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`
- El servidor automáticamente pasa los archivos `.php` al interprete de PHP.
- Se puede configurar el servidor de tal manera que haga lo mismo con archivos `.html` pero eso no es necesario.
- ¿Por qué a algunos desarrolladores les interesa pasar `.html` por el interprete de PHP? .

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`
- El servidor automáticamente pasa los archivos `.php` al interprete de PHP.
- Se puede configurar el servidor de tal manera que haga lo mismo con archivos `.html` pero eso no es necesario.
- ¿Por qué a algunos desarrolladores les interesa pasar `.html` por el interprete de PHP? esconder el hecho de usar PHP.

Introducción a PHP

- Durante las proximas 3 o 4 semanas nos enfocaremos en aprender PHP.
- Extensión `.php`
- El servidor automáticamente pasa los archivos `.php` al interprete de PHP.
- Se puede configurar el servidor de tal manera que haga lo mismo con archivos `.html` pero eso no es necesario.
- ¿Por qué a algunos desarrolladores les interesa pasar `.html` por el interprete de PHP? esconder el hecho de usar PHP.
- Un archivo `.php` finalmente terminará enviando al navegador código HTML.

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.
- Ejemplo:

```php
<?php
  echo "Hello world";
?>
```

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.
- Ejemplo:

```
<?php
  echo "Hello world";
?>
```

- La manera de utilizar las etiquetas es flexible.

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.
- Ejemplo:

```
<?php
  echo "Hello world";
?>
```

- La manera de utilizar las etiquetas es flexible.
  - Se puede abrir la etiqueta al inicio del documento y cerrarla al final. En este caso todo el HTML es escrito directamente desde PHP. Menor complejidad y mayor legibilidad.

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.
- Ejemplo:

```
<?php
  echo "Hello world";
?>
```

- La manera de utilizar las etiquetas es flexible.
  - Se puede abrir la etiqueta al inicio del documento y cerrarla al final. En este caso todo el HTML es escrito directamente desde PHP. Menor complejidad y mayor legibilidad.
  - Se puede abrir múltiples etiquetas en el documento y dentro de estas se escribe solo pequeños fragmentos de PHP. El resto es HTML puro. Más rápido (ligeramente), quizás un poco más complejo por el hecho de abrir y cerrar varias etiquetas.

Las etiquetas PHP

- Para empezar a escribir código PHP necesitamos la etiqueta: `<?php`
- Para terminar de escribir PHP usamos la etiqueta de cierre `?>`.
- Ejemplo:

```php
<?php
  echo "Hello world";
?>
```

- La manera de utilizar las etiquetas es flexible.
  - Se puede abrir la etiqueta al inicio del documento y cerrarla al final. En este caso todo el HTML es escrito directamente desde PHP. Menor complejidad y mayor legibilidad.
  - Se puede abrir múltiples etiquetas en el documento y dentro de estas se escribe solo pequeños fragmentos de PHP. El resto es HTML puro. Más rápido (ligeramente), quizás un poco más complejo por el hecho de abrir y cerrar varias etiquetas.
- El estilo que elijamos es preferencia personal.

Variación de etiquetas

- Ocasionalmente encontraremos (en Internet) que el código PHP está dentro de las etiquetas <? ?>

```
<?
  echo "Hello world";
?>
```

Variación de etiquetas

- Ocasionalmente encontraremos (en Internet) que el código PHP está dentro de las etiquetas <? ?>

```
<?
  echo "Hello world";
?>
```

- Aunque son aceptadas y funcionan, no se recomiendan por incompatibilidad. Esta opción será removida en futuras versiones de PHP.

Etiqueta de cierre es opcional

- Cuando tenemos solamente código PHP en nuestro archivo, se puede omitir la etiqueta de cierre ?>.

Etiqueta de cierre es opcional

- Cuando tenemos solamente código PHP en nuestro archivo, se puede omitir la etiqueta de cierre ?>.
- Es es incluso recomendado y una buena práctica ya que nos aseguramos que no existen espacios es blanco después del código PHP lo que generaría errores cuando se utiliza código orientado a objetos.

Contenido I

Contenido I

Comentarios

- Existen dos formas de añadir comentarios:

```
// Este un comentario de una linea
```

Comentarios

- Existen dos formas de añadir comentarios:

```
// Este un comentario de una linea
```

- Útil para comentar lineas específicas de código:

```
// echo "X es igual a $x";
```

Comentarios

- Existen dos formas de añadir comentarios:

```
// Este un comentario de una linea
```

- Útil para comentar lineas específicas de código:

```
// echo "X es igual a $x";
```

- O para añadir información sobre una línea de código:

```
$x += 10; // Incrementar $x por 10
```

Comentarios

- Existen dos formas de añadir comentarios:

```
// Este un comentario de una linea
```

- Útil para comentar lineas específicas de código:

```
// echo "X es igual a $x";
```

- O para añadir información sobre una línea de código:

```
$x += 10; // Incrementar $x por 10
```

- Otra manera de comentar es con /* */

```
<?php
/* Esta es una seccion
   de varias lineas
   que esta comentada */
?>
```

Comentarios

- Existen dos formas de añadir comentarios:

```
// Este un comentario de una linea
```

- Útil para comentar lineas específicas de código:

```
// echo "X es igual a $x";
```

- O para añadir información sobre una línea de código:

```
$x += 10; // Incrementar $x por 10
```

- Otra manera de comentar es con /* */

```
<?php
/* Esta es una seccion
   de varias lineas
   que esta comentada */
?>
```

- Común para comentar grandes fragmentos de código.

Comentarios

- Los comentarios deben ser utilizados a lo largo del documento PHP para clarificar el código y brindar más información cuando es necesario.

Comentarios

- Los comentarios deben ser utilizados a lo largo del documento PHP para clarificar el código y brindar más información cuando es necesario.
- Los comentarios no pueden ser anidados. Es decir no podemos comentar código que ya contiene los caracteres `/* */`, esto nos dará error.

Contenido I

Sintaxis básica

- PHP es un lenguaje simple, tiene sus raíces en C y Perl aunque se ve más como Java.

Sintaxis básica

- PHP es un lenguaje simple, tiene sus raíces en C y Perl aunque se ve más como Java.
- Es flexible, pero se deben seguir algunas reglas de sintaxis y estructura.

Sintaxis básica

- PHP es un lenguaje simple, tiene sus raíces en C y Perl aunque se ve más como Java.
- Es flexible, pero se deben seguir algunas reglas de sintaxis y estructura.
- Toda sentencia termina con (;). Ej.

```
$x += 10;
```

Sintaxis básica

- PHP es un lenguaje simple, tiene sus raíces en C y Perl aunque se ve más como Java.
- Es flexible, pero se deben seguir algunas reglas de sintaxis y estructura.
- Toda sentencia termina con (;). Ej.

```
$x += 10;
```

- Cuando no se termina con ; PHP interpretará todo como una sola sentencia. Al no ser entendido mostrará un mensaje `Parse error`.

El símbolo $

- Este símbolo ($) se pone antes de cada variable.

El símbolo $

- Este símbolo ($) se pone antes de cada variable.
- Permite interpretar el lenguaje más rápido.

```php
<?php
  $mycounter = 1;
  $mystring = "Hello";
  $myarray = array("One", "Two", "Three");
?>
```

Recomendaciones

- Usar identación en el código.

Recomendaciones

- Usar identación en el código.
- Los espacios en blanco hacen que el código sea más legible. Ej.

```php
<?php
if($x<=10){
   $a=3;
   $b=4;
   $i++;
}
?>

<?php
if( $x <= 10 ){
   $a = 3;
   $b = 4;
   $i++;
}
?>
```

Ejemplo PHP

Crear el archivo `prueba1.php`, escribir el siguiente programa y visualizarlo en el navegador:

```php
<?php // test1.php
  $username = "Fred Smith";
  echo $username;
  echo "<br>";
  $current_user = $username;
  echo $current_user;
?>
```

Contenido I

Variables String

- Se puede ver las variables de PHP como una caja donde almacenamos un pedazo de papel con un valor (numérico, texto, etc) escrito en el.

```
$username = "Fred Smith";
```

Variables String

- Se puede ver las variables de PHP como una caja donde almacenamos un pedazo de papel con un valor (numérico, texto, etc) escrito en el.

```
$username = "Fred Smith";
```

- Las comillas (“”) nos indican que estamos almacenando un string (cadena de caracteres). Se puede utilizar (“”) o (‘’). La diferencia la veremos luego.

Variables String

- Se puede ver las variables de PHP como una caja donde almacenamos un pedazo de papel con un valor (numérico, texto, etc) escrito en el.

```php
$username = "Fred Smith";
```

- Las comillas (“”) nos indican que estamos almacenando un string (cadena de caracteres). Se puede utilizar (“”) o (‘’). La diferencia la veremos luego.
- Para ver el contenido de la variable:

```php
echo $username;
```

Variables String

- Se puede ver las variables de PHP como una caja donde almacenamos un pedazo de papel con un valor (numérico, texto, etc) escrito en el.

```php
$username = "Fred Smith";
```

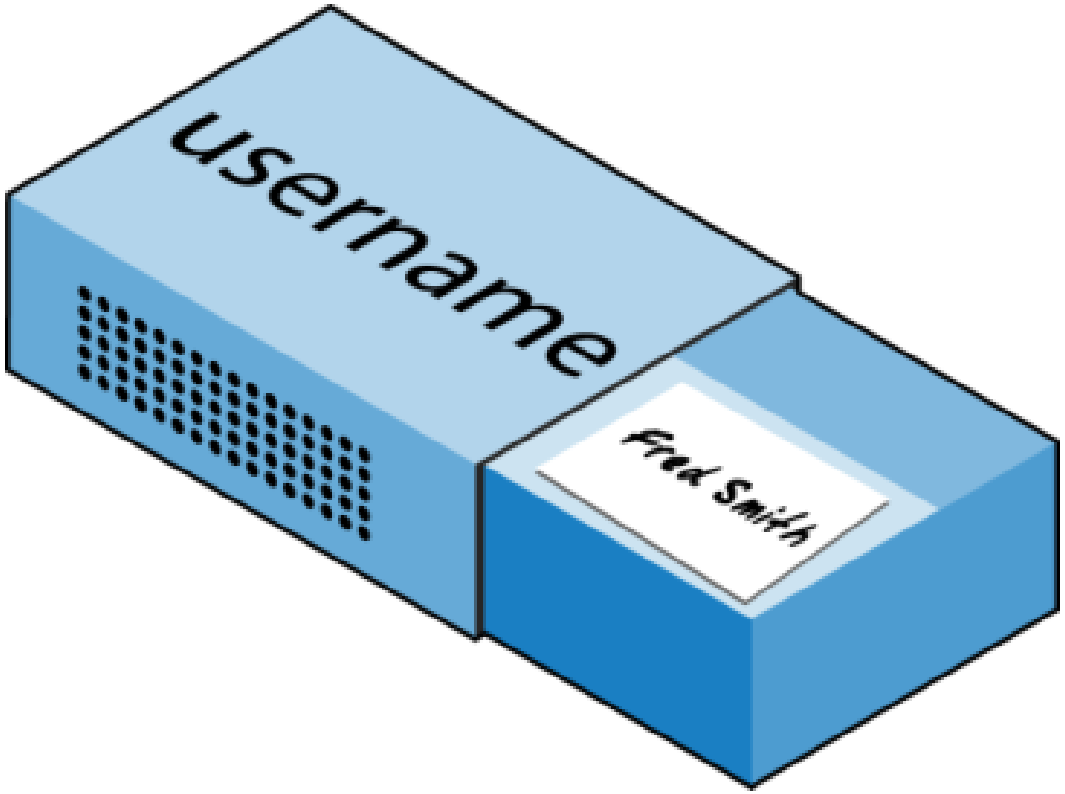
- Las comillas (“”) nos indican que estamos almacenando un string (cadena de caracteres). Se puede utilizar (“”) o (‘’). La diferencia la veremos luego.
- Para ver el contenido de la variable:

```php
echo $username;
```

- Podemos asignar el valor de la variable a otra variable:

```php
$current_user = $username;
```

Una caja de fósforos



Variables numéricas

```php
<?php
$count = 17;
$count = 17.5;
echo $count;
?>
```

Arreglos

- Podemos ver los arreglos con varias caja de fósforos pegadas unas a otras.

```
$team = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

Arreglos

- Podemos ver los arreglos con varias caja de fósforos pegadas unas a otras.

```php
$team = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

- La sintaxis de `array` es: `array();`

Arreglos

- Podemos ver los arreglos con varias caja de fósforos pegadas unas a otras.

```php
$team = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

- La sintaxis de array es: `array();`
- Si quisieramos sber el elemento en la posición 4:

```php
echo $team[3]; // Imprime el nombre Chris
```

Arreglos

- Podemos ver los arreglos con varias caja de fósforos pegadas unas a otras.

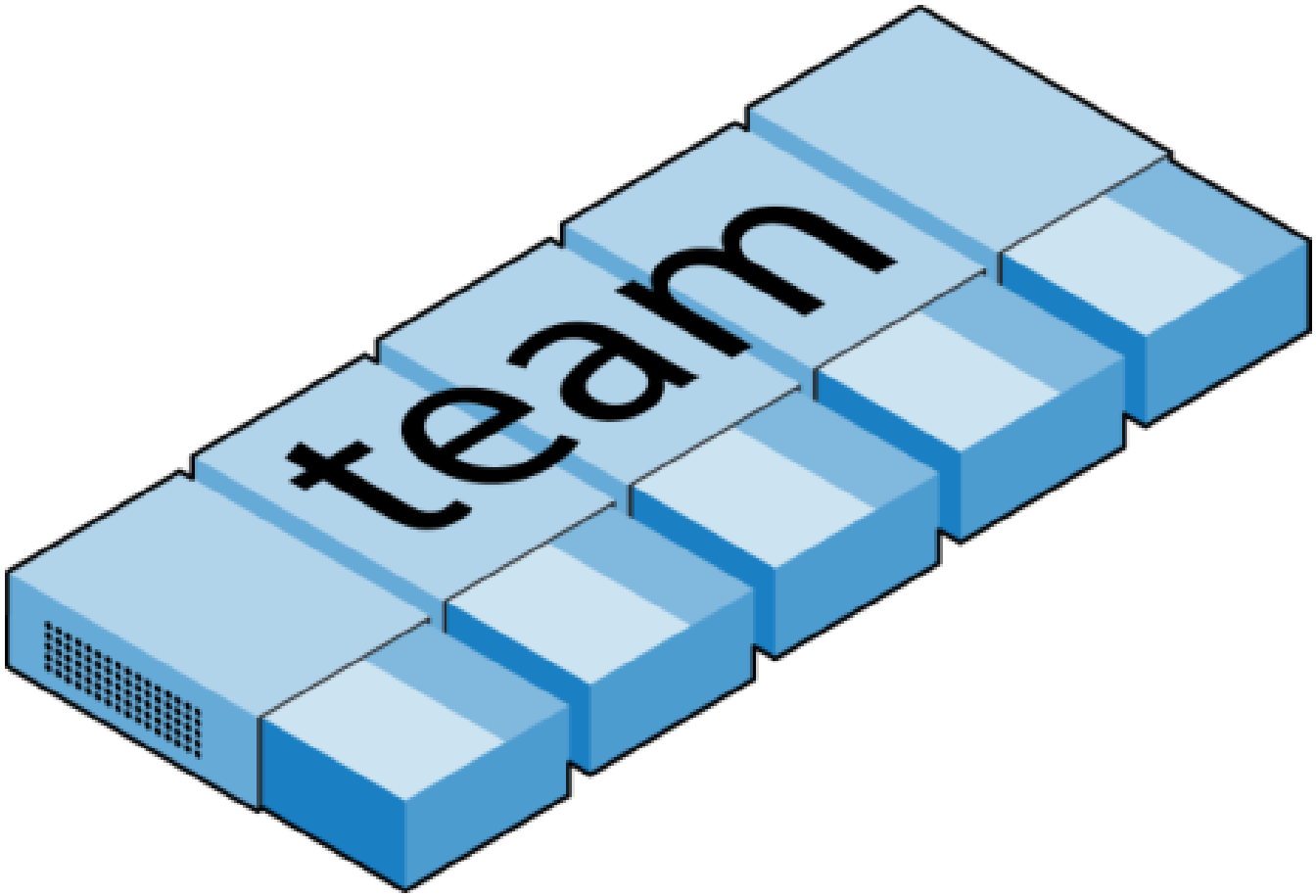
```php
$team = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

- La sintaxis de `array` es: `array();`
- Si quisieramos sber el elemento en la posición 4:

```php
echo $team[3]; // Imprime el nombre Chris
```

- El primer elemento es el 0.

Arrays



Arrays multi-dimensionales

- Los arrays en PHP nos permiten hacer muchas cosas.

Arrays multi-dimensionales

- Los arrays en PHP nos permiten hacer muchas cosas.
- Los arrays pueden ser multidimensionales (bi, tri o más)

Arrays multi-dimensionales

- Los arrays en PHP nos permiten hacer muchas cosas.
- Los arrays pueden ser multidimensionales (bi, tri o más)
- Supongamos que queremos implementar un juego de tres en raya.

Arrays multi-dimensionales

- Los arrays en PHP nos permiten hacer muchas cosas.
- Los arrays pueden ser multidimensionales (bi, tri o más)
- Supongamos que queremos implementar un juego de tres en raya.
- Necesitamos 9 celdas en una matriz de 3x3. Un arreglo de dos dimensiones.

Array bidimensional

Con arreglo con un juego de tres en raya en progreso.

```php
<?php
  $oxo = array(array('x', ' ', 'o'),
  array('o', 'o', 'x'),
  array('x', 'o', ' '));
?>
```

- Dos dimensiones: `array(array());`.

Array bidimensional

Con arreglo con un juego de tres en raya en progreso.

```php
<?php
  $oxo = array(array('x', ' ', 'o'),
  array('o', 'o', 'x'),
  array('x', 'o', ' '));
?>
```

- Dos dimensiones: `array(array());`.
- Para acceder al tercer elemento de la segunda fila:

```php
echo $oxo[1][2];
```

Array bidimensional

Con arreglo con un juego de tres en raya en progreso.

```php
<?php
  $oxo = array(array('x', ' ', 'o'),
  array('o', 'o', 'x'),
  array('x', 'o', ' '));
?>
```

- Dos dimensiones: `array(array());`.
- Para acceder al tercer elemento de la segunda fila:

```php
echo $oxo[1][2];
```

- Más adelante veremos más detalles con `arrays()`.

Reglas para nombres de variables

Los nombres de variables:

- Empiezan con una letra del abecedario o guión bajo (_).

Reglas para nombres de variables

Los nombres de variables:

- Empiezan con una letra del abecedario o guión bajo (_).
- Contienen solamente caracteres de la a-z, A-Z, 0-9 y _

Reglas para nombres de variables

Los nombres de variables:

- Empiezan con una letra del abecedario o guión bajo (_).
- Contienen solamente caracteres de la a-z, A-Z, 0-9 y _
- No aceptan espacios. Las palabras se pueden separar con _

Reglas para nombres de variables

Los nombres de variables:

- Empiezan con una letra del abecedario o guión bajo (_).
- Contienen solamente caracteres de la a-z, A-Z, 0-9 y _
- No aceptan espacios. Las palabras se pueden separar con _
- Son sensibles a mayúsculas y minúsculas (case-sensitive). `$User_Name` no es igual que `$user_name`.

Reglas para nombres de variables

Los nombres de variables:

- Empiezan con una letra del abecedario o guión bajo (_).
- Contienen solamente caracteres de la a-z, A-Z, 0-9 y _
- No aceptan espacios. Las palabras se pueden separar con _
- Son sensibles a mayúsculas y minúsculas (case-sensitive). `$User_Name` no es igual que `$user_name`.
- Los acentos son aceptados pero es recomendable evitarlos para aumentar la compatibilidad con otros programadores.

Contenido I

Operadores

- Todos los comandos matemáticos, string, de comparación y lógicos (+, -, *, /).

Operadores

- Todos los comandos matemáticos, string, de comparación y lógicos (+, -, *, /).
- `echo 6 + 2; //8`

Operadores

- Todos los comandos matemáticos, string, de comparación y lógicos (+, -, *, /).
- `echo 6 + 2; //8`
- Operadores Aritméticos

Operator	Description	Example
+	Addition	$j + 1
-	Subtraction	$j - 6
*	Multiplication	$j * 11
/	Division	$j / 4
%	Modulus (division remainder)	$j % 9
++	Increment	++$j
--	Decrement	--$j

Operadores ++ / --

- ¿Cuál es la salida de la siguiente sentencia, si $x = 9?

```
if (++$x == 10) echo $x;
```

Operadores ++ / --

- ¿Cuál es la salida de la siguiente sentencia, si $x = 9?

```
if (++$x == 10) echo $x;
```

- ¿Y de esta, si $y = 0?

```
if ($y-- == 0) echo $y;
```

Operadores de asignación

Operator	Example	Equivalent to
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Operadores de comparación

Operator	Description	Example
==	Is *equal* to	`$j == 4`
!=	Is *not equal* to	`$j != 21`
>	Is *greater than*	`$j > 3`
<	Is *less than*	`$j < 100`
>=	Is *greater than or equal* to	`$j >= 15`
<=	Is *less than or equal* to	`$j <= 8`

Operadores lógicos

Operator	Description	Example
&&	*And*	`$j == 3 && $k == 2`
and	Low-precedence *and*	`$j == 3 and $k == 2`
||	*Or*	`$j < 5 || $j > 10`
or	Low-precedence *or*	`$j < 5 or $j > 10`
!	*Not*	`! ($j == $k)`
xor	*Exclusive or*	`$j xor $k`

Concatenación de strings

- Para concatenar strings utilizamos en punto (.)

```
echo ``Usted tiene '' . $msgs . `` mensajes.'';
```

Concatenación de strings

- Para concatenar strings utilizamos en punto (.)

```
echo ``Usted tiene '' . $msgs . `` mensajes.'';
```

- Si asumimos que `$msgs` tiene un valor de 5, la salida será: `Usted tiene 5 mensajes.`

Concatenación de strings

- Para concatenar strings utilizamos en punto (.)

```
echo ``Usted tiene '' . $msgs . `` mensajes.'';
```

- Si asumimos que `$msgs` tiene un valor de 5, la salida será: `Usted tiene 5 mensajes.`
- Se puede utilizar también el operador: `.=`

```
$boletin .= $noticias;
```

Tipos de string

- Existen dos tipos de string en PHP. Se lo diferencia por el tipo de comillas: ‘’ o “”

Tipos de string

- Existen dos tipos de string en PHP. Se lo diferencia por el tipo de comillas: ‘’ o “”
- Si utilizamos comillas simples (‘’) el valor del string será literal (contenido exacto). No intentará evaluar las variables dentro del string.

```
$info = 'Las variables llevan un prefijo $ asi:
    $variable';
```

Tipos de string

- Existen dos tipos de string en PHP. Se lo diferencia por el tipo de comillas: ‘’ o “”
- Si utilizamos comillas simples (‘’) el valor del string será literal (contenido exacto). No intentará evaluar las variables dentro del string.

```
$info = 'Las variables llevan un prefijo $ asi:
    $variable';
```

- Si queremos incluir el valor de una variable en un string, utilizamos comillas dobles (“”). Esto facilita la concatenación de string con variables. *sustitución de variables*.

```
echo "Esta semana, $count personas han visto tu perfil";
```

Caracteres de escape

- Para poder utilizar ciertos caracteres con algún significado para PHP necesitamos utilizar el caracter de escape (\). Ej. la siguiente línea dará error:

```
$text = 'La comilla simple es ' esto dara error.'; //
    error de sintaxis
```

Caracteres de escape

- Para poder utilizar ciertos caracteres con algún significado para PHP necesitamos utilizar el caracter de escape (\). Ej. la siguiente línea dará error:

```
$text = 'La comilla simple es ' esto dara error.'; //
    error de sintaxis
```

- Deberiamos escibirlo así:

```
$text = 'La comilla simple es \' esto es correcto.'; //
    correcto
```

Caracteres de escape

- Para poder utilizar ciertos caracteres con algún significado para PHP necesitamos utilizar el caracter de escape (\). Ej. la siguiente línea dará error:

```
$text = 'La comilla simple es ' esto dara error.'; //
    error de sintaxis
```

- Deberiamos escibirlo así:

```
$text = 'La comilla simple es \' esto es correcto.'; //
    correcto
```

- Lo mismo se puede aplicar a las comillas dobles:

```
$text = "La comilla doble es \" esto es correcto."; //
    correcto
```

Caracteres de escape

- Para poder utilizar ciertos caracteres con algún significado para PHP necesitamos utilizar el caracter de escape (\). Ej. la siguiente línea dará error:

```
$text = 'La comilla simple es ' esto dara error.'; //
    error de sintaxis
```

- Deberiamos escibirlo así:

```
$text = 'La comilla simple es \' esto es correcto.'; //
    correcto
```

- Lo mismo se puede aplicar a las comillas dobles:

```
$text = "La comilla doble es \" esto es correcto."; //
    correcto
```

- Los caracteres \t, \n, \r solo pueden ser interpretados utilizando (“”). Las comillas simples lo presentarán literal. Las comillas simples solo interpretarán los caracteres \’ y \\

Contenido I

Comandos de múltiples líneas

- Podemos escribir texto entre comillas en múltiples líneas

```php
<?php
  $author = "Steve Ballmer";
  echo "Developers, Developers, developers, developers,
  developers, developers, developers, developers!

  - $author.";
?>
```

Comandos de múltiples líneas

- Podemos escribir texto entre comillas en múltiples líneas

```php
<?php
  $author = "Steve Ballmer";
  echo "Developers, Developers, developers, developers,
  developers, developers, developers, developers!

  - $author.";
?>
```

- O asignar a un variable:

```php
<?php
  $author = "Bill Gates";
  $text = "Measuring programming progress by lines of
      code is like
  Measuring aircraft building progress by weight.

  - $author.";
?>
```

Operador <<< (*Heredoc*)

```php
<?php
$author = "Brian W. Kernighan";
echo <<<_END
Debugging is twice as hard as writing the code in the first
    place.
Therefore, if you write the code as cleverly as possible,
    you are,
by definition, not smart enough to debug it.

- $author.
_END;
?>
```

- PHP imprime todo literalmente, incluyendo espacios, y tabulaciones sin necesidad de añadir \n.
- La etiqueta de cierre solo debe ir acompañada de (;) ningún espacio en blanco adicional.
- También se puede asignar a variables.

Contenido I

Tipos de variables

- PHP es muy flexible con respecto a los tipos.

Tipos de variables

- PHP es muy flexible con respecto a los tipos.
- Las variables no tienen que ser declaradas antes de ser usadas.

Tipos de variables

- PHP es muy flexible con respecto a los tipos.
- Las variables no tienen que ser declaradas antes de ser usadas.
- PHP siempre convierte las variables al *tipo* requerido dependiendo del contexto.

```php
<?php
  $number = 12345 * 67890; // numerico
  echo substr($number, 3, 1); // string
?>

<?php
  $pi = "3.1415927"; //string
  $radius = 5; //numerico
  echo $pi * ($radius * $radius); // numerico
?>
```

Contenido I

Constantes

- Similares a las variables pero su contenido no se puede modificar.

Constantes

- Similares a las variables pero su contenido no se puede modificar.
- Una constante comúnmente declarada es el directorio de nuestro *document root*:

```
define("ROOT_LOCATION", "/usr/local/www/");
```

Constantes

- Similares a las variables pero su contenido no se puede modificar.
- Una constante comúnmente declarada es el directorio de nuestro *document root*:

```php
define("ROOT_LOCATION", "/usr/local/www/");
```

- Para leer el valor:

```php
$directory = ROOT_LOCATION;
echo $directory;
echo ROOT_LOCATION;
```

Constantes

- Similares a las variables pero su contenido no se puede modificar.
- Una constante comúnmente declarada es el directorio de nuestro *document root*:

```php
define("ROOT_LOCATION", "/usr/local/www/");
```

- Para leer el valor:

```php
$directory = ROOT_LOCATION;
echo $directory;
echo ROOT_LOCATION;
```

- No llevan el prefijo $ y solo pueden ser definidas con `define();` una sola vez.

Constantes mágicas

Constante mágica	Descripción
__LINE__	Linea actual del archivo.
__FILE__	El directorio completo (incluyendo nombre del archivo) del archivo actual.
__DIR__	El directorio del archivo.
__FUNCTION__	El nombre de la función actual (case-sensitive).
__CLASS__	El nombre de la clase (case-sensitive).
__METHOD__	El nombre del método (case-sensitive)
__NAMESPACE__	El nombre del namespace (case-sensitive).

```php
echo "This is line " . __LINE__ . " of file " . __FILE__;
```

Contenido I

echo vs print

- Su función es similar.

`echo` vs `print`

- Su función es similar.
- `print` es una especie de función que toma un parámetro y retorna siempre 1.

`echo` vs `print`

- Su función es similar.
- `print` es una especie de función que toma un parámetro y retorna siempre 1.
- `echo` es parte del lenguaje PHP.

`echo` vs `print`

- Su función es similar.
- `print` es una especie de función que toma un parámetro y retorna siempre 1.
- `echo` es parte del lenguaje PHP.
- Ninguno necesita paréntesis.

`echo` vs `print`

- Su función es similar.
- `print` es una especie de función que toma un parámetro y retorna siempre 1.
- `echo` es parte del lenguaje PHP.
- Ninguno necesita paréntesis.
- `echo` es más rápido porque no retorna valores.

echo vs print

- Su función es similar.
- `print` es una especie de función que toma un parámetro y retorna siempre 1.
- `echo` es parte del lenguaje PHP.
- Ninguno necesita paréntesis.
- `echo` es más rápido porque no retorna valores.
- `echo` no puede usarse en expresiones mas complejas, `print` si puede.

```
$b ? print "TRUE" : print "FALSE";
```

Contenido I

Funciones

- Utilizadas para separar secciones de código con tareas distintas.

Funciones

- Utilizadas para separar secciones de código con tareas distintas.
- Reutilización de código (con todos sus beneficios).

Funciones

- Utilizadas para separar secciones de código con tareas distintas.
- Reutilización de código (con todos sus beneficios).
- Reciben parámetros y retornan valores.

```php
<?php
  function longdate($timestamp)
  {
    return date("l F jS Y", $timestamp);
  }

  echo longdate(time());
  echo longdate(time() - 17 * 24 * 60 * 60); // la fecha
       hace 17 dias (17 d, 24 h, 60 m,60 s)
?>
```

Contenido I

Alcance de variables

- Las variables se pueden definir como locales o globales.

Alcance de variables

- Las variables se pueden definir como locales o globales.
- Por defecto, las variables creadas dentro de una función son **locales** a esa función y las variables creadas fuera de cualquier función son accesibles solo fuera de cualquier función.

```php
<?php
  function longdate($timestamp)
  {
    $temp = date("l F jS Y", $timestamp);
    return "The date is $temp";
  }
?>
```

Ejemplo (fallará)

```php
<?php
  // MOSTRARA UNA NOTA DICIENDO QUE LA VARIABLE NO HA SIDO
      DEFINIDA
  $temp = "The date is ";
  echo longdate(time());
  function longdate($timestamp)
  {
    return $temp . date("l F jS Y", $timestamp);
  }
?>
```

Ejemplo (Dos alternativas de solución)

```php
<?php
  $temp = "The date is ";
  echo $temp . longdate(time());
  function longdate($timestamp)
  {
    return date("l F jS Y", $timestamp);
  }
?>
```

O

```php
<?php
  $temp = "The date is ";
  echo longdate($temp, time());
  function longdate($text, $timestamp)
  {
    return $text . date("l F jS Y", $timestamp);
  }
?>
```

Variables globales

- Cuando necesitamos que sea accedida desde cualquier parte del código.

Variables globales

- Cuando necesitamos que sea accedida desde cualquier parte del código.
- Cuando una variable contiene datos grandes y complejos y no se desea pasarla como argumento a varias funciones.

```
global $is_logged_in;
```

Variables globales

- Cuando necesitamos que sea accedida desde cualquier parte del código.
- Cuando una variable contiene datos grandes y complejos y no se desea pasarla como argumento a varias funciones.

```
global $is_logged_in;
```

- Se debe ser cuidadoso para no cambiar el valor de la variable accidentalmente o utilizar el mismo nombre para otra variable.

Variables globales

- Cuando necesitamos que sea accedida desde cualquier parte del código.
- Cuando una variable contiene datos grandes y complejos y no se desea pasarla como argumento a varias funciones.

```
global $is_logged_in;
```

- Se debe ser cuidadoso para no cambiar el valor de la variable accidentalmente o utilizar el mismo nombre para otra variable.
- Se recomienda que los nombre de las variables globales estén en mayúsculas para evitar confusiones.

Variables estáticas

- Sirven para mantener el valor de las variables a través de varias las llamadas a una función.

Variables estáticas

- Sirven para mantener el valor de las variables a través de varias las llamadas a una función.
- Ejemplo: deseamos mantener el valor de un contador en varias llamadas a la función:

```php
<?php
  function test()
  {
    static $count = 0; // solo ejecutada primera llamada
    echo $count;
    $count++;
  }
?>
```

Variables estáticas

- Sirven para mantener el valor de las variables a través de varias las llamadas a una función.
- Ejemplo: deseamos mantener el valor de un contador en varias llamadas a la función:

```php
<?php
  function test()
  {
    static $count = 0; // solo ejecutada primera llamada
    echo $count;
    $count++;
  }
?>
```

- Solo aceptan valores predefinidos (no el resultado de otra expresión).

```php
<?php
  static $int = 0; // Permitido
  static $int = 1+2; // No permitido (Parse error)
  static $int = sqrt(144); // No Permitido
?>
```

Variables superglobales

- Son variables predefinidas en el lenguaje y son accesibles desde cualquier parte.

Variables superglobales

- Son variables predefinidas en el lenguaje y son accesibles desde cualquier parte.
- Contienen información sobre el programa que está corriendo y su ambiente de ejecución.

Variables superglobales

- Son variables predefinidas en el lenguaje y son accesibles desde cualquier parte.
- Contienen información sobre el programa que está corriendo y su ambiente de ejecución.
- Definidas como arrays asociativos (que veremos luego).

Variables superglobales

Nombre	Contenido
$GLOBALS	Todas las variables definidas como globales. Los nombres son los índices.
$_SERVER	Info sobre headers, paths y ubicación de scripts. Creados por el servidor web puede variar.
$_GET	Variables pasadas al script por el método HTTP Get.
$_POST	Variables pasadas al script por el método HTTP Post.
$_FILES	Items subidos al script por el método HTTP Post
$_COOKIE	Variables pasadas al script por via HTTP cookies
$_SESSION	Variables de sesión en el script actual
$_REQUEST	Contenidos de información pasados por el navegador ($_GET, $_POST y $_COOKIE)
$_ENV	Variables pasadas al script por el método "environment"

Como usarlas?

```
print_r ($_SERVER);
$came_from = $_SERVER['HTTP_REFERER'];
```

Superglobales (seguridad)

- Se debe ser cuidadoso con las superglobales ya que son utilizadas por hackers para romper seguridades en el sitio y acceder a información sensible.

Superglobales (seguridad)

- Se debe ser cuidadoso con las superglobales ya que son utilizadas por hackers para romper seguridades en el sitio y acceder a información sensible.
- Se pueden cargar comandos Unix o MySQL en $_POST o $_GET.

Superglobales (seguridad)

- Se debe ser cuidadoso con las superglobales ya que son utilizadas por hackers para romper seguridades en el sitio y acceder a información sensible.
- Se pueden cargar comandos Unix o MySQL en $_POST o $_GET.
- Siempre se debe limpiar las superglobales antes de usarlas.

Superglobales (seguridad)

- Se debe ser cuidadoso con las superglobales ya que son utilizadas por hackers para romper seguridades en el sitio y acceder a información sensible.
- Se pueden cargar comandos Unix o MySQL en $_POST o $_GET.
- Siempre se debe limpiar las superglobales antes de usarlas.
- Se puede utilizar la función `htmlentities()` que convierte todos los caracteres en entidades HTML (< = < y > = >).

```php
$came_from = htmlentities($_SERVER['HTTP_REFERER']);
```

Superglobales (seguridad)

- Se debe ser cuidadoso con las superglobales ya que son utilizadas por hackers para romper seguridades en el sitio y acceder a información sensible.
- Se pueden cargar comandos Unix o MySQL en $_POST o $_GET.
- Siempre se debe limpiar las superglobales antes de usarlas.
- Se puede utilizar la función `htmlentities()` que convierte todos los caracteres en entidades HTML (< = < y > = >).

```php
$came_from = htmlentities($_SERVER['HTTP_REFERER']);
```

- `htmlentities` debe usarse siempre que se procese información enviada por los usuarios y que vaya a ser presentada (no solo con superglobales).