Server side processing: Introducción a PHP

Área de Ingeniería Telemática

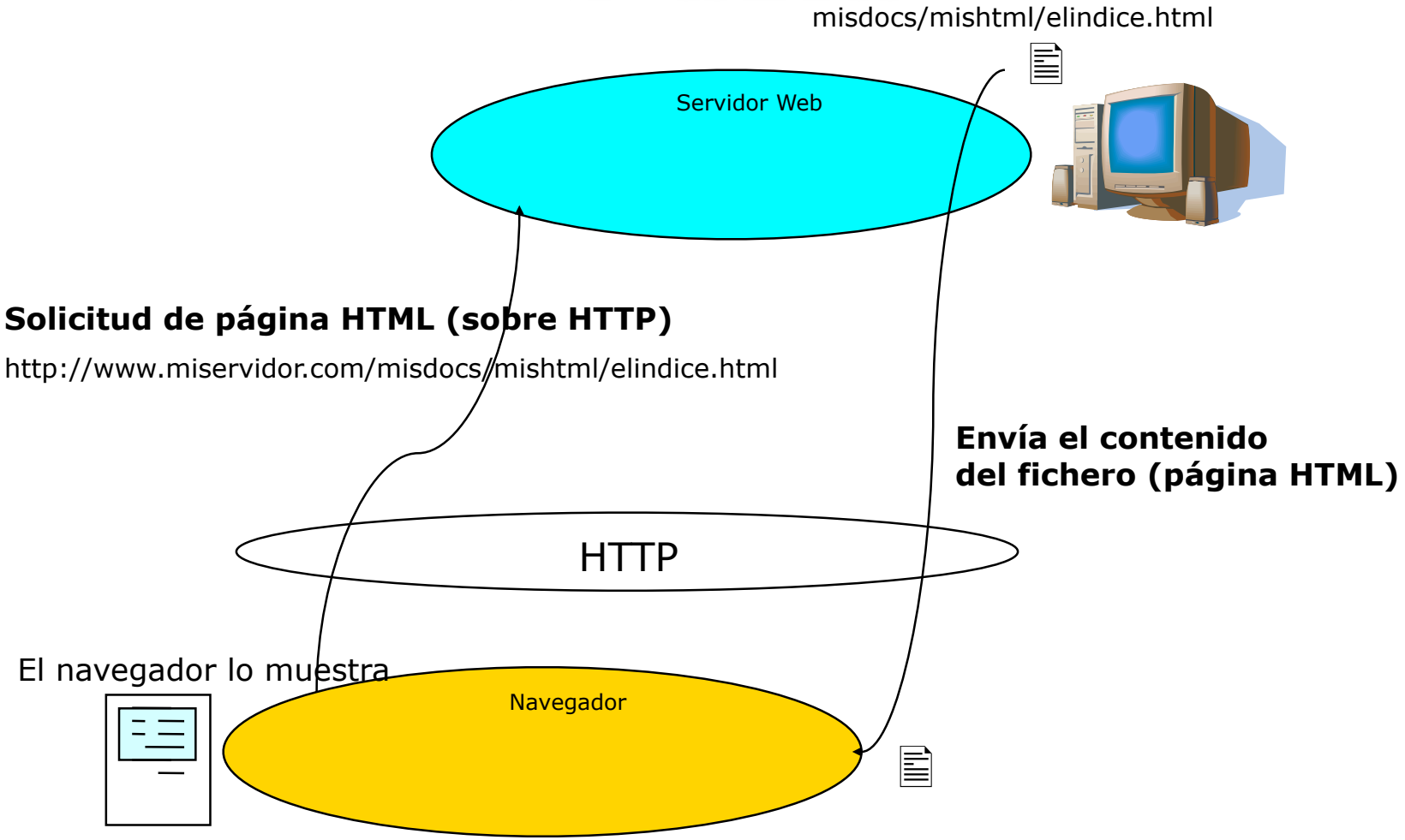
Contenido

- HTML estático -> interactividad -> aplicación Web
- Server side processing en PHP
  - Introducción
  - Sintaxis básica
  - Tipos
  - Variables
  - Operadores
  - Estructuras de control
  - Funciones
  - Alcance de las variables
  - Superglobals
  - Constantes

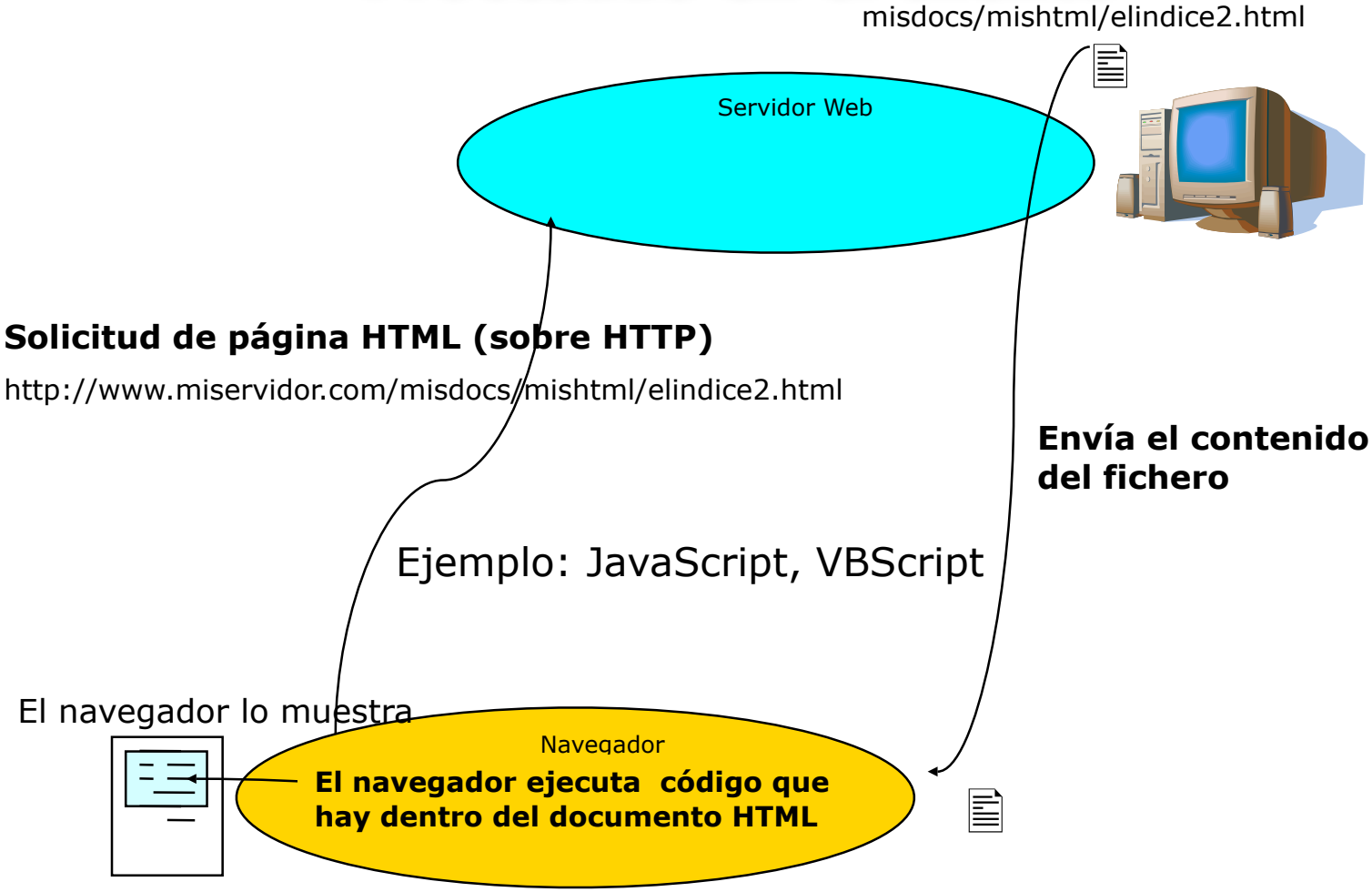
De HTML estático a aplicación Web

- Inicialmente
  - Servidor almacena paginas Web: texto enriquecido con enlaces a otros elementos
  - Interactividad solo en el orden que elijo para leerlos
- Descripción de formularios
  - Puedo enviar información al servidor
  - Y si el servidor utiliza esa información para generar nuevas páginas
  - ¿Como funciona un libro de visitas o los comentarios de un blog?
    - ¿Se modifica el fichero de la pagina Web añadiendo los comentarios?
- Se puede generar una pagina Web en el momento para mi (i.e. que ocurre si pido ver una webcam??)
  - ¿Cómo de rápido? (i.e. googlemaps)

HTML "estático"

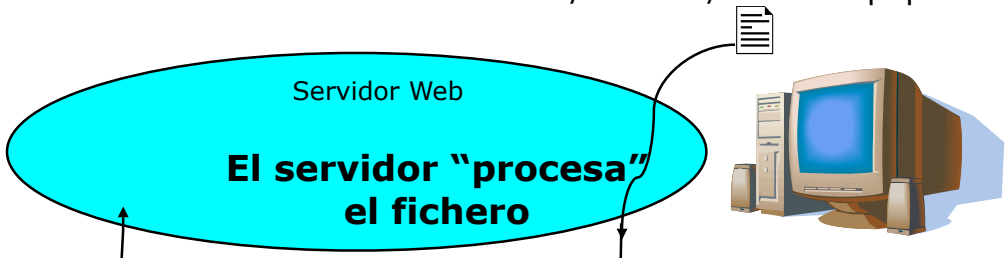


Procesado en el cliente



Procesado en el servidor

misdocs/mishtml/elindice3.php



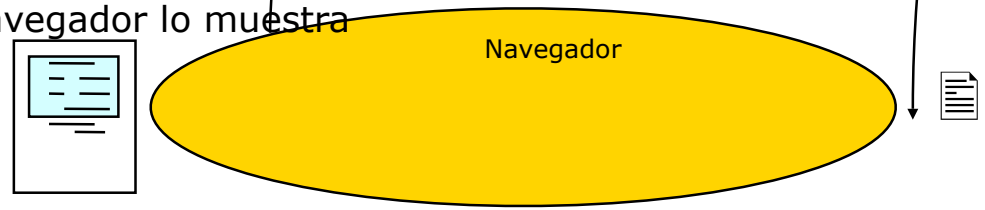
Solicitud de página HTML (sobre HTTP)

http://www.miservidor.com/misdocs/mishtml/elindice3.php

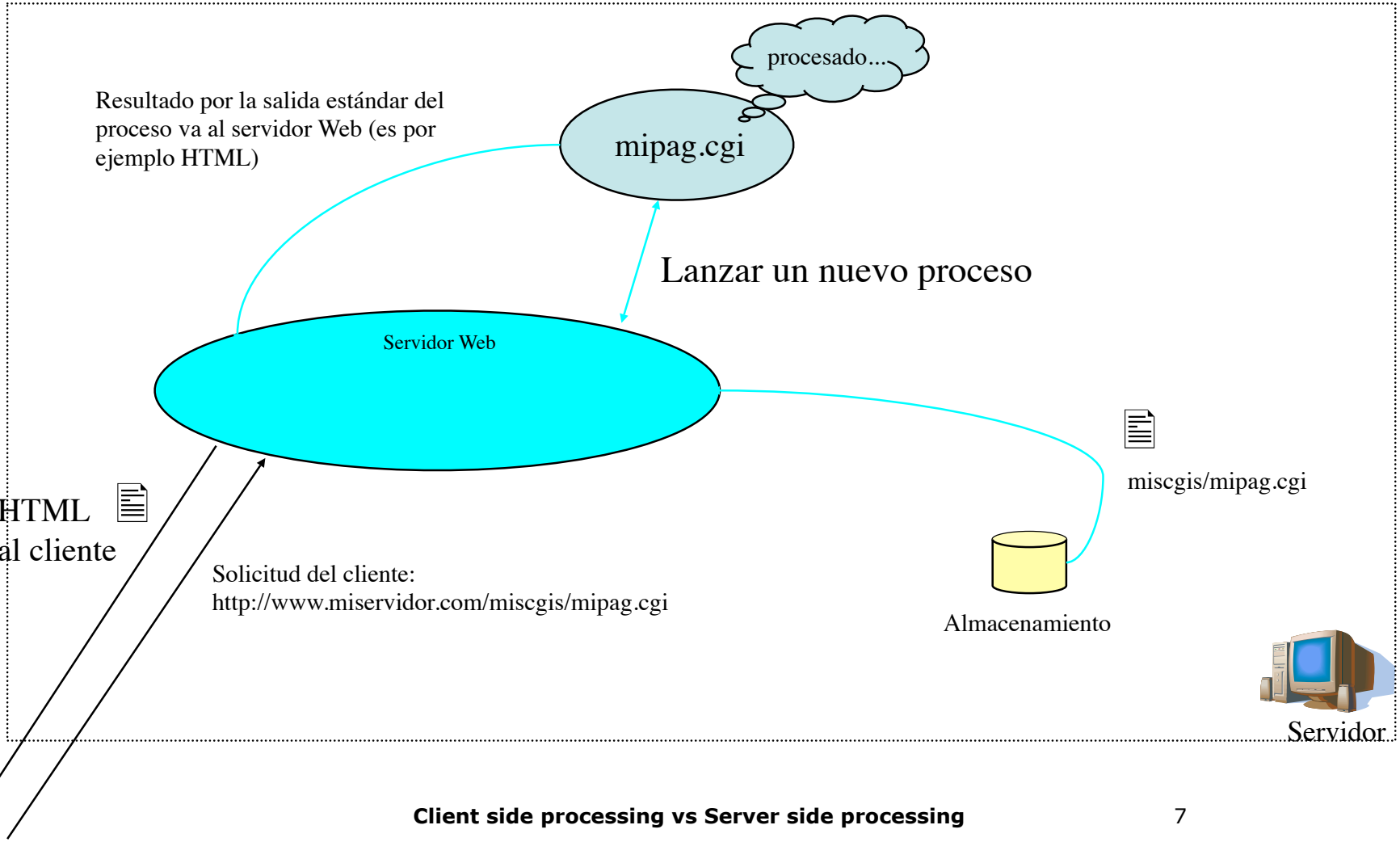
Envía el resultado del procesamiento

Ejemplo: PHP, CGIs, ASP, JSP

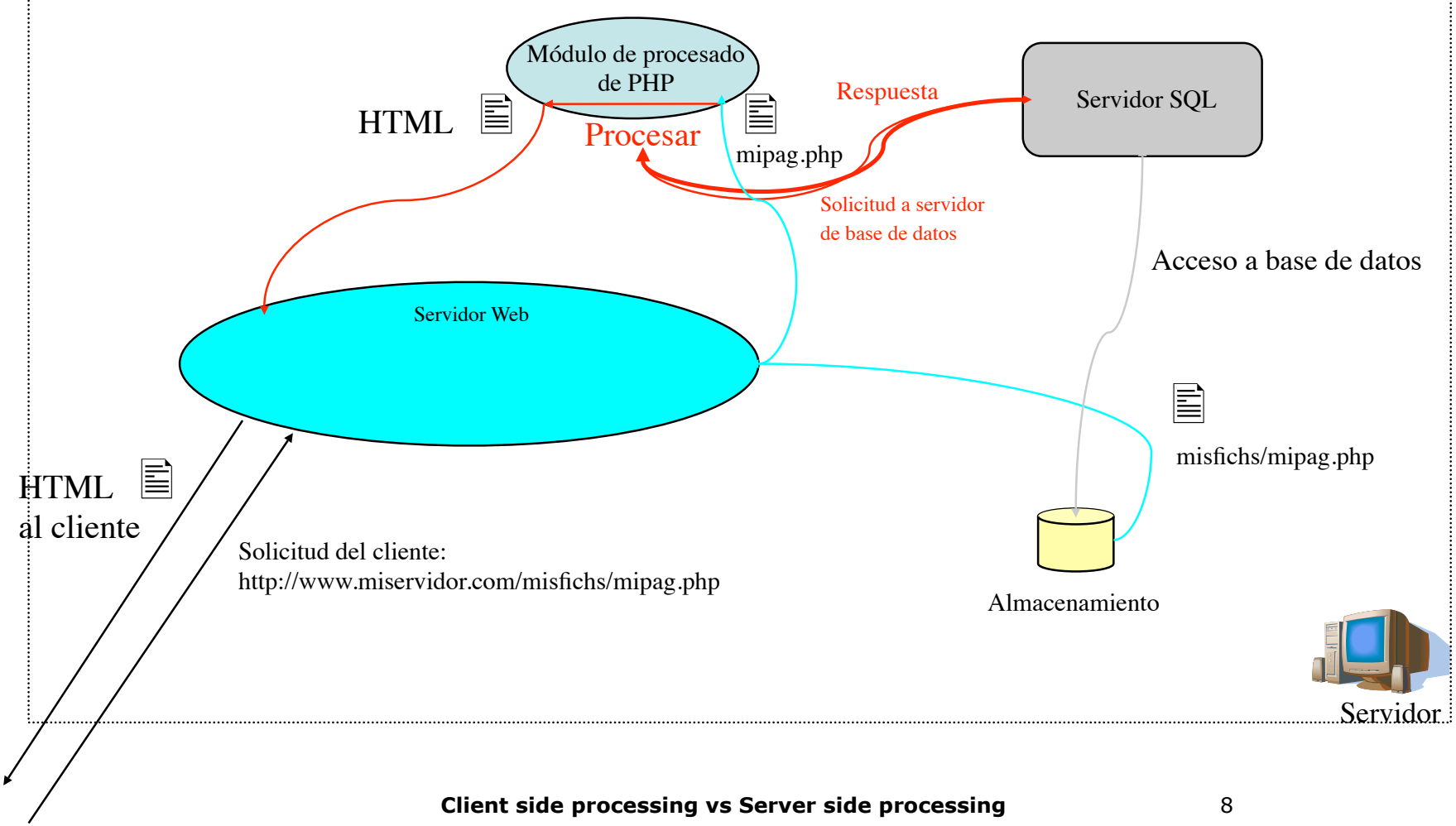
El navegador lo muestra



Ejemplo: procesado en servidor (CGI)



Ejemplo: procesado en servidor (PHP+base de datos)



Server-side processing

- Generando páginas en el momento para el usuario
  - Generar contenido en el momento
  - Leer la información que llega del usuario
  - Leer o guardar información en una base de datos
  - Guardar el estado y reconocer al usuario
  - ...
- El lenguaje PHP...

¿Qué es PHP?

- PHP = "**PHP Hypertext Preprocessor**"
- Lenguaje de scripts para el servidor Web (server-side processing)
- Open Source
- Puede ir en el mismo documento que el código HTML
- Simple para el principiante
- Con muchas características avanzadas
- Soportado en gran número de sistemas operativos: variantes de UNIX (Linux, HP-UX, Solaris, OpenBSD), Microsoft Windows, Mac OS X
- Soporta la mayoría de servidores web: Apache, Microsoft IIS, Personal Web Server, iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, etc
- Permite no solo generar HTML sino también imágenes, PDFs, Flash, XML …
- Soporta un gran número de bases de datos: Oracle, mSQL, MySQL, ODBC, Sybase, etc
- Soporta un gran número de servicios y protocolos: LDAP, IMAP, SNMP, NNTP, POP3, HTTP, etc

¿Puede ir en el documento HTML?

- Dentro del documento HTML podemos emplear tags especiales que indican que lo que va entre ellos es código PHP. Ejemplo:

ejemplo.php

```
<html>
  <head><title>Script de ejemplo</title></head>
  <body>
    <h1>Pagina simple</h1>
    <p>Aqui el codigo HTML</p>
    <?php
      print "<p>Y esto sale del código PHP</p>\n";
    ?>
    <p>Has visto el párrafo anterior?</p>
  </body>
</html>
```

Código PHP

- En el servidor se establece que ficheros pueden contener código PHP. Generalmente por la extensión del fichero
- El servidor busca en el documento los tags que marcan el código PHP. Lo ejecuta y si el script quiere escribir texto (print() o echo()) ese texto aparece donde estaba el código PHP al enviarse el documento (no se cambia el fichero)

¿Puede ir en el documento HTML?

- Dentro del documento HTML podemos emplear tags especiales que indican que lo que va entre ellos es código PHP. Ejemplo:

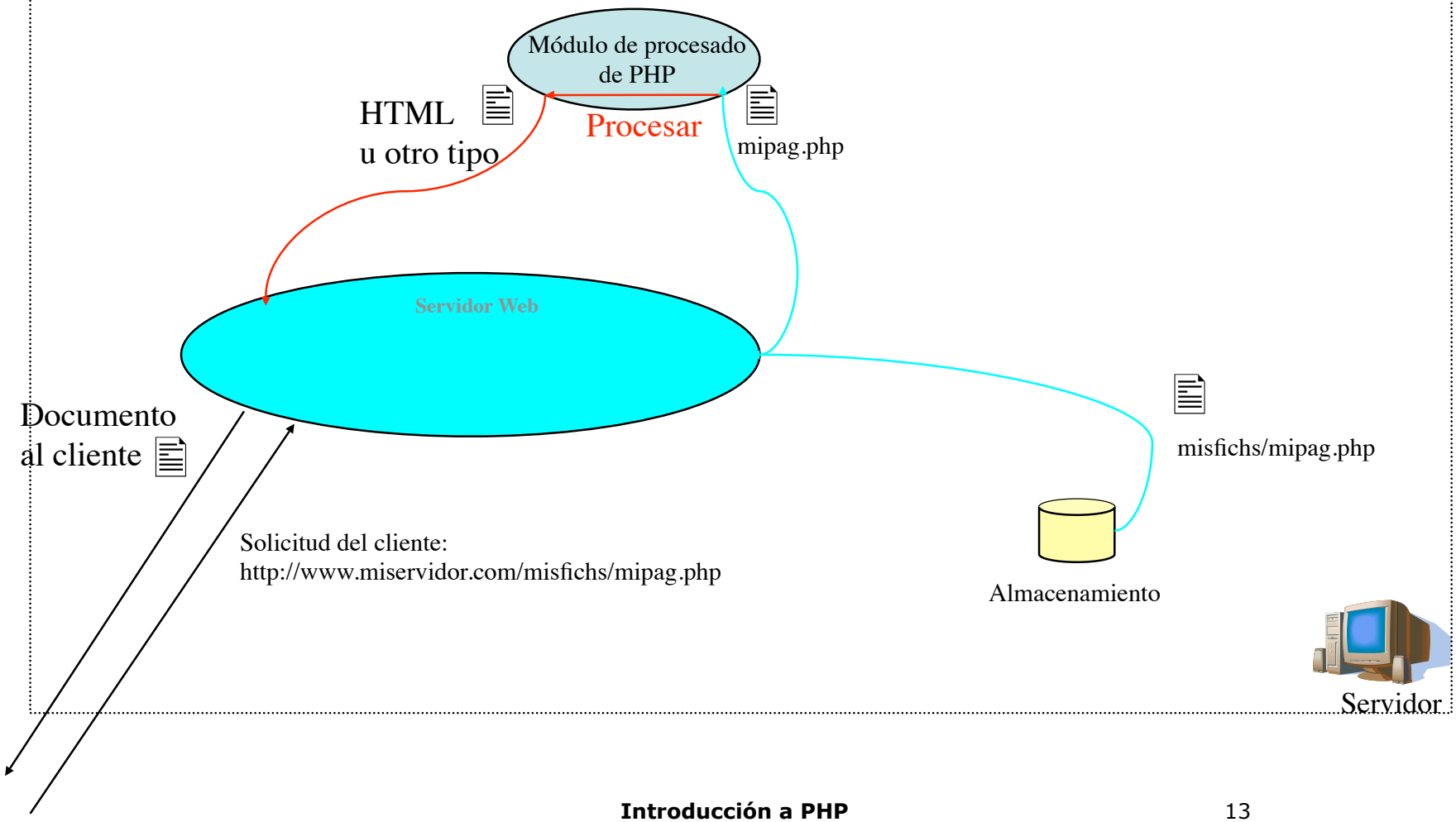
ejemplo.php

```
<html>
  <head><title>Script de ejemplo</title></head>
  <body>
    <h1>Pagina simple</h1>
    <p>Aqui el codigo HTML</p>

    <p>Y esto sale del código PHP</p>

    <p>Has visto el párrafo anterior?</p>
  </body>
</html>
```

Código PHP

- En el servidor se establece que ficheros pueden contener código PHP. Generalmente por la extensión del fichero
- El servidor busca en el documento los tags que marcan el código PHP. Lo ejecuta y si el script quiere escribir texto (print() o echo()) ese texto aparece donde estaba el código PHP al enviarse el documento (no se cambia el fichero)

Procesado en servidor



Sintaxis básica

- El módulo de PHP busca uno de los tags que emplea para reconocer el comienzo de código PHP
- Ejecuta el código hasta encontrar una marca de final de código
- Continúa por el documento hasta encontrar otra marca de comienzo
- Todo lo que esté fuera de esas marcas queda inalterado
- Las formas más adecuadas (siempre disponibles) de marcar el comienzo y final de código PHP son:

```
<?php          ?>
<script language="php">      </script>
```

- Las sentencias terminan en **;** aunque la última no lo necesita
- Comentarios estilo C++ (`/* */` o `//`) o estilo Shell (`#`)
- Se puede "salir de modo PHP" dentro de condiciones, por ejemplo:

```
<?php if ($expression) {
?>
    <strong>This is true.</strong>
<?php
                    }
 else {
?>
    <strong>This is false.</strong>
<?php
                    }
?>
```

Tipos

- `boolean`
  - Dos valores posibles: `true` y `false` (case-insensitive)
  - Muchos operadores devuelven un valor `boolean` que se puede emplear en una estructura de control

- `integer`
  - Se pueden representar en base 8, 10 ó 16 de igual forma que en C
  - El tamaño en bits depende de la plataforma
  - Si se desborda se convierte en un `float`
  - No existe operador de división entera

Tipos

- `float`
  - Se pueden introducir en formato decimal o mantisa-exponente
  - Su tamaño en bits depende de la plataforma

- `string`
  - Es una serie de caracteres (bytes)
  - No hay límite para su tamaño
  - Cadenas literales se pueden expresar mediante:
    - Apóstrofes o comas simples (`'`)
      - El apóstrofe se hace aparecer escapándolo (`\'`)
      - En su interior no se interpretan variables u otras secuencias escapadas (por ejemplo ignora `\n`)

Tipos

- Dobles comillas (")
  - En su interior sí interpreta variables y secuencias escapadas típicas (`\n`, `\t`, etc)
- Heredoc
  - Se puede indicar que todo lo que venga a continuación es una cadena hasta encontrar un cierto texto. Ejemplo

```
echo <<< FIN
Ejemplo de string
que ocupa múltiples líneas
FIN
```

  - El texto funciona como texto entre dobles comillas

Tipos

- Se puede acceder a un carácter en concreto con `$cadena{indice}` donde el índice es un entero que empieza en 0 (también es válido `$cadena[indice]`)
- Se pueden concatenar con el operador **.**
  - Ejemplo: $string3 = $string1.$string2;
- Otro tipo se convierte a cadena automáticamente cuando el contexto lo requiere (por ejemplo en un `print()`)
- Una cadena se convierte automáticamente en el número que contiene en el comienzo de su texto cuando el contexto lo requiere

Tipos

- `array`
  - En realidad contienen un mapeo entre *claves* y *valores* (array asociativo)
  - El mismo array se puede emplear como array asociativo o como array indexado
  - Los elementos pueden ser de cualquier tipo (incluso otros arrays) y ser de tipos diferentes
  - Se crean con `array()`

```
array( clave => valor, ...)
ejemplo: $unarray = array("dia" => 15, 1 => "uno");
ejemplo: $otro = array("unarray" => array(0=>14,
  4=>15), "nombre" => "Una tabla");
```

  - La clave puede ser un entero o un string

Tipos

- En una asignación al array, si no se especifica la clave/ índice:
  - Se emplea como índice el máximo índice entero +1
  - Si no hay ningún índice entero se emplea 0
- En las asignaciones se emplean corchetes para indicar la clave/índice

```
$miarray['nombre']="Daniel";
```

- Si el array no existe se crea
- Para eliminar un elemento del array hay que emplear `unset()`

```
unset($miarray['nombre']);
unset($miarray);
```

Tipos

- `object`
  - Tiene que existir una clase para el objeto
  - PHP4 soporta clases (incluidas "con calzador") y sólo tiene métodos públicos. PHP5 sí soporta desde la primera versión del código
  - Las clases se definen con `class`
    - ejemplo:

```
class Carro {
  var $contenido;
  function add_uno($nombre){
          $this->contenido[]=$nombre;
  }
}
```

Tipos

- Las clases soportan herencia (no múltiple) empleando extends.
  - ejemplo: `class OtroCarro extends Carro {}`
- Los objetos se crean con `new`
  - ejemplo: `$micarro=new Carro;`
- Se accede a variables o métodos con `->`
  - ejemplo: `$micarro->add_uno("Libro de Stevens");`
- Se puede convertir el objeto en una cadena con `serialize()` y de nuevo en un objeto con `unserialize()` (útil para guardar en fichero, sistemas de cache, etc)

Tipos

- `resource`
  - Contiene una referencia a algún recurso externo (por ejemplo la conexión a una base de datos)
  - Son creados por funciones especiales

- **Funciones útiles:**
  - `var_dump()`: Muestra información sobre una variable (tipo, contenido), incluyendo la estructura y contenido de arrays
  - `print_r()`: Muy similar pero muestra también propiedades protegidas o privadas

Variables

- Tipos: `boolean, integer, float, string, array, object, resource`
- No hay que declarar las variables
- Se pueden declarar empleando la palabra reservada `var` (independiente del tipo): `var $mivariable;`
- PHP decide en tiempo de ejecución el tipo que deben tener según el contexto en que se empleen
- Antes del nombre de la variable hay que poner siempre `$`
- Su nombre distingue mayúsculas de minúsculas
- Nombre: `'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'`
- A partir de PHP 4 se puede asignar una variable a otra por valor o por referencia:

  Por valor: `$mivar="Jose";` `$mivar=$miotravar;`

  Por referencia: `$tercera=&$mivar;`

  Más info: http://es2.php.net/manual/en/language.references.php

Operadores

- Aritméticos: `+ - * / %`
- Asignación: `= += -= *= /= %= .=`
- Operaciones sobre bits: `& | ^ ~ << >>`
- Comparación: `== === != <> !== < > <= >= ?:`
- Control de errores: `@`
- Ejecución: `` `` `` (equivalente a system(), exec(), etc)
  ejemplo: `$listado=`ls -l`;`
- Incremento/Decremento: ++ -- (prefijos y postfijos)
- Lógicos: `and or xor ! && ||`
- Cadenas: `. .=`
- Arrays: `+` (une dos arrays, si hay claves duplicadas se mantienen los valores del array de la izquierda)

Estructuras de control

- if (expresion) {} elseif (expresion) {} else {}
- while (expresion) {}
- do {} while (expresion);
- for (expre1; expr2; expr3) {}
- foreach => Para recorrer arrays

  Ejemplo:

```
$arr = array("uno" => "one", "dos" => "two", "tres" =>
"three");
foreach ($arr as $valor) {
    echo "Valor: $valor<br>\n";
}
```

Estructuras de control

Ejemplo:

```
$arr = array("uno" => "one", "dos" => "two", "tres" =>
  "three");
foreach ($arr as $clave => $valor) {
  echo "Clave: $clave Valor: $valor<br>\n";
}
```

- break
- continue
- switch (expresion) { case expresion: }
- return => sale de funciones, del script global o de scripts incluidos
- include()
  - Permite "incluir" otro script php en ese lugar (similar a #include en C)

Estructuras de control

  - Al interpretar ese otro fichero empieza en modo HTML así que si el contenido es PHP tendremos que marcarlo (`<?php ?>`)
  - Se puede poner dentro de una condición (entonces es obligatorio poner {}). Ejemplo:

```
if ($condicion) {
  include $pathfichero;
}
```

- require(): Análogo a include pero ante fallos (no encontrar el fichero) `include()` da un `Warning` y `require()` un `Fatal Error`
- include_once y requiere_one: realiza un include o un require de un fichero y sólo una única vez

Funciones

- Ejemplo:

```
function mifuncion($arg1, $arg2){
  echo "Esta es mi funcion y me has pasado $arg1 y $arg2";
  return 33;
}
```

- Dentro de la función puede aparecer cualquier código válido PHP. Eso incluye otras funciones, clases, etc.
- No hace falta que la función esté definida antes de la línea donde se emplee (PHP 4)
- En el nombre de la función no distingue mayúsculas
- Se pueden definir dentro de un bloque de condición y entonces no existen hasta que la ejecución pase por él

Funciones

- Los argumentos pueden tener valores por defecto (y como en C++ no debe haber argumentos a su derecha sin valor por defecto).
  - Ejemplo:

```
function mifunc2($arg1, $arg2="no me han pasado nada")
```

Alcance de las variables

- Variables definidas en el script (globales) alcanzan al contenido de ficheros incluidos (`include`, `require`)
- Dentro de las funciones no se ven las variables externas sino solo las que se declaren ahí
- Para que una variable externa se pueda emplear dentro de una función hay que declararla en ella como global. Ejemplo:

```
$a="Hola";
function mifunc(){
  global $a;
  print $a;
}
```

Alcance de las variables

- Se pueden declarar variables estáticas (como en C) con `static` que dentro de funciones mantendrán el valor de una llamada a la función a otra.

  Particularidades: http://es.php.net/static
- Existen unas variables llamadas *Superglobales* que existen en cualquier punto del script (incluso dentro de funciones sin declararlas como globales)

Superglobals

- No definidas por el usuario
- Disponibles en cualquier punto
- `$GLOBALS`
  - Un array con referencias a las variables globales en ese punto (la clave es el nombre de la variable)
- `$_SERVER`
  - Variables declaradas por el servidor web (array con los nombres de las variables como claves)

    `$_SERVER['SERVER_NAME']`

    `$_SERVER['QUERY_STRING']` ....etc...
- `$_GET`
  - Array con las variables enviadas por el navegador mediante el método `GET`

Superglobals

- `$_POST`
  - Idem mediante el método `POST`
- `$_COOKIE`
  - Idem mediante cookies
- `$_FILES`
  - Array con las variables proporcionadas al script por medio de la subida de ficheros via HTTP (POST)
- `$_ENV`
  - Array con variables proporcionadas al script por medio del entorno
- `$_SESSION`
  - Array con variables registradas en la sesión del script

Constantes

- Pueden contener un `boolean`, `integer`, `float` o `string`
- Su nombre distingue mayúsculas de minúsculas pero por convenio se suelen usar solo mayúsculas
- Su alcance es global (como las superglobals)
- Se crean con: `define(constante, valor);`

  ejemplo: `define("MICONSTANTE", "Hola mundo.");`
- Se emplean poniendo su nombre (sin un `$` delante)
- También se puede acceder al contenido de una constante empleando: `constant(constante);`
- Una vez creadas no se puede cambiar su contenido ni eliminarlas

Resumen

- Dentro de páginas HTML
- Sencillo
- Sintaxis parecida a C
- Clases, pero no orientado a objetos
  - PHP 5 incluye un nuevo modelo de objetos
- Un gran número de opciones avanzadas