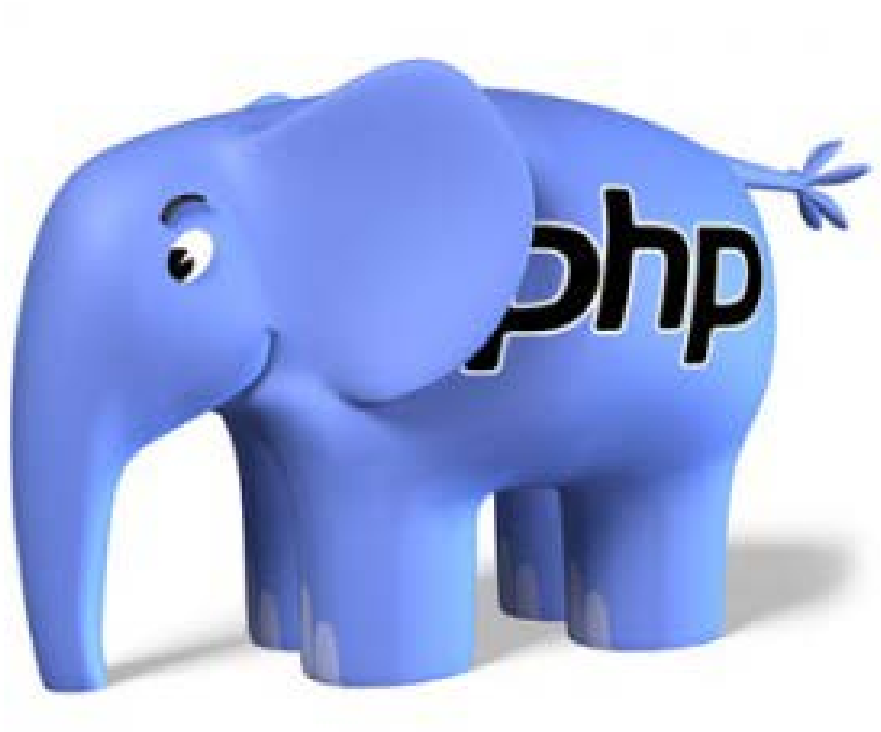
Desarrollo de software con PhP

Jonathan Leonardo Silva Blasio
bls.jonathan.proyectos@gmail.com

https://www.edmodo.com/home#/join/ep7kry
Código: 2en94n

PHP



¿Qué es PhP?

- Es un lenguaje de programación de script interpretado del lado del servidor por **Zend Engine**.
- Permite generar páginas Web dinámicas, al ser **embebido en HTML**.
- Usado actualmente por más de 200 millones de sitios Web.
- Ampliamente documentado.
- Open source
- Multiplataforma
- Creado por **Rasmus Lerdorf** en 1995
- Originalmente significaba Personal Home Page Tools
- PHP ahora significa: PHP Hypertext Preprocessor
- http://www.php.net

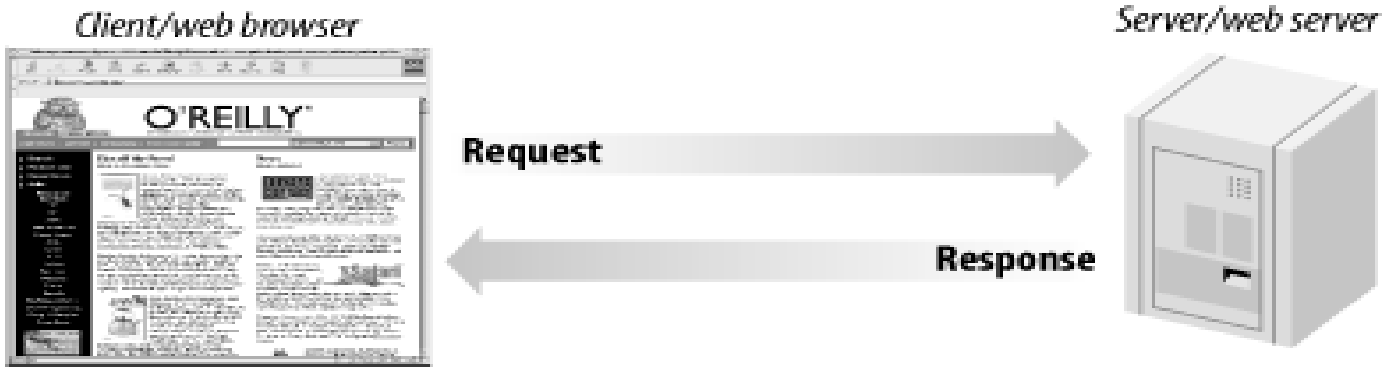
Caracteristicas

- **Código fuente abierto**: permite posibles mejoras o sugerencias acerca de su funcionamiento.
- **Gratuito**: no es necesario realizar ningún desembolso económico.
- **Portable y multiplataforma**: versiones para múltiples plataformas (Windows 98, NT, 2000, 7, 8, 10, Unix, Linux, etc.).
- **Eficiente**: consume muy pocos recursos en el servidor.
- **Alta velocidad de desarrollo**: Proporciona gran cantidad de librerías muy útiles y bien documentadas que ahorran mucho trabajo al programador.
- Permite las técnicas de **Programación Orientada a Objetos**.

Comunicación entre el navegador y el servidor

Se lleva a cabo en dos etapas:

1. El navegador (cliente) envía una **solicitud** a un servidor Web.
2. El servidor Web atiende la solicitud y envía la **respuesta** de regreso al cliente.

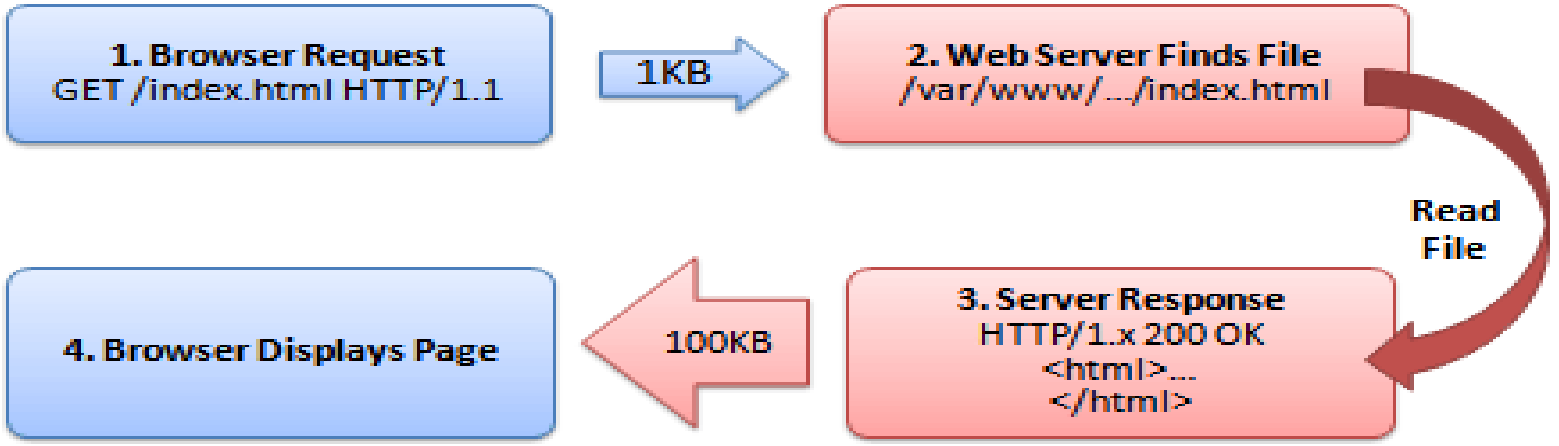


Protocolo HTTP

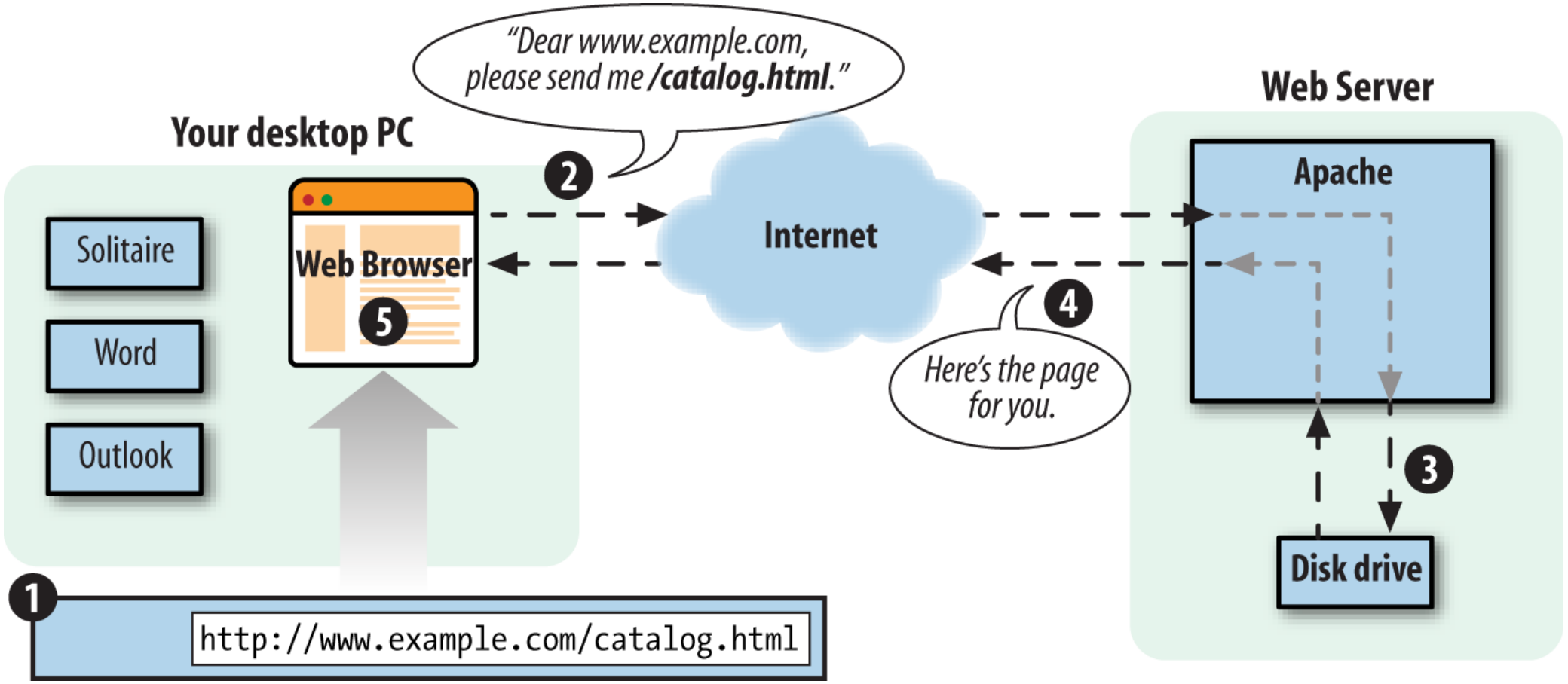
- El protocolo HTTP (Hypertext Transfer Protocol) está diseñado para habilitar la **comunicación entre clientes y servidores**.
- HTTP funciona a base de **peticiones y respuestas** entre un cliente y un servidor.
- El **navegador** funciona como cliente y el **servidor** puede ser cualquier computadora que cuente con el software para transmisión de documentos (servidor Web).

¿Cómo funciona?

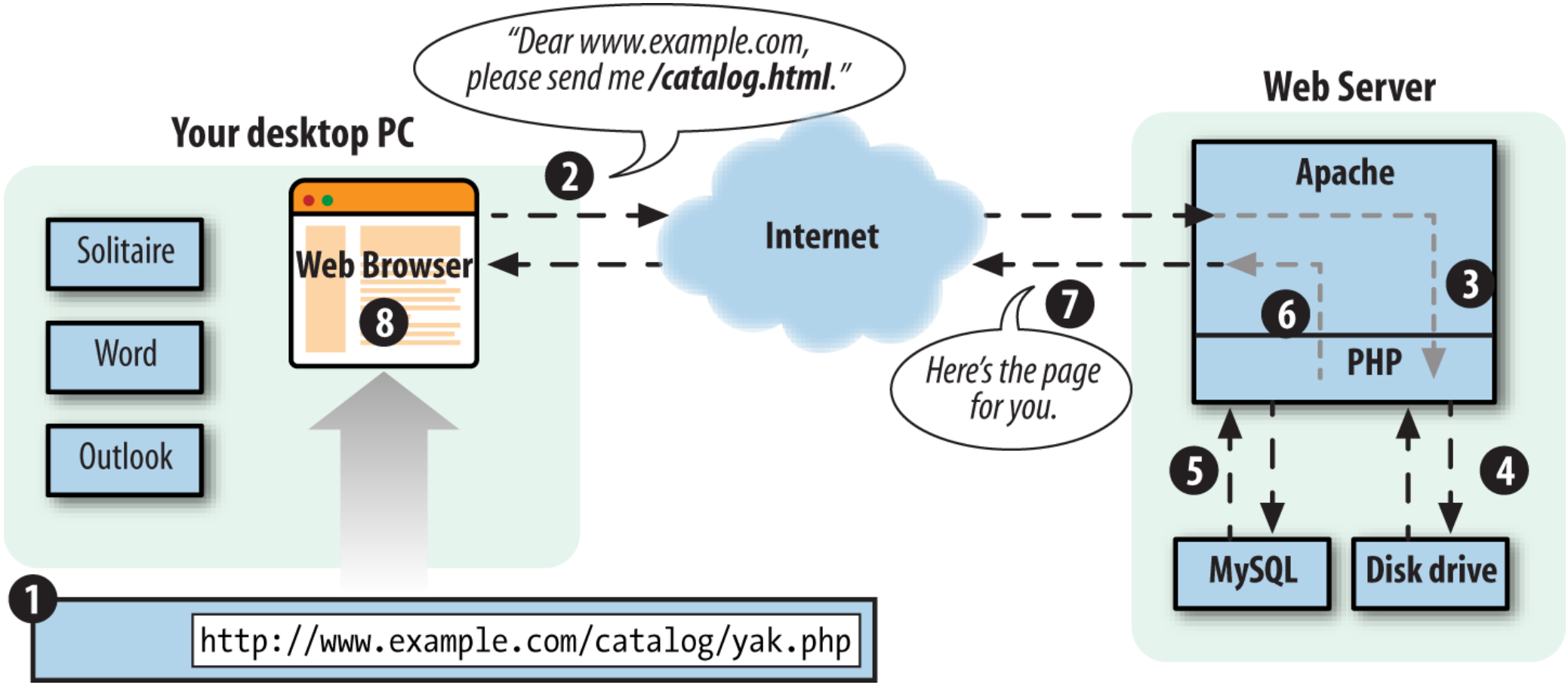
- Para consultar una página, el navegador envía una **petición** al Servidor.
- Después el servidor regresa una **respuesta** al cliente.



Paginas web estáticas



Paginas web dinámicas



Sintaxis básica

- PHP es un lenguaje embebido en HTML. (El código PHP se inserta dentro del HTML)
- Es necesario indicarle al intérprete cuáles son las partes escritas en PHP.
- Los bloques de código PHP deben comenzar con **<?php** y terminar con **?>**
- Dentro de un bloque de código puede haber una o varias instrucciones.
- Cada sentencia o instrucción debe terminar con punto y coma.

Ejemplo

```
<html>
  <head>
    <title>Mi primer página PHP</title>
  </head>
  <body>
    <?php echo "Hola mundo";?>
  </body>
</html>
```

Sintaxis básica

```php
<?php
    //Comentario Simple
    /**
     * Comentario
     * multiples lineas
    */
    echo "Hola mundo";
?>
```

Desarrollo de software con PhP

Jonathan Leonardo Silva Blasio
bls.jonathan.proyectos@gmail.com

https://www.edmodo.com/home#/join/ep7kry
Código: 2en94n

Tipos de datos

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- Booleanos
- Enteros
- Números de Punto Flotante
- Cadenas

Dos tipos compuestos:

- Arreglos
- Objetos

Y finalmente dos tipos especiales:

- Recursos
- Nulo

Booleanos

- Los valores booleanos expresan un valor de verdad, el cual puede ser únicamente

TRUE o FALSE.

- Nota: las palabras **true** y **false** se pueden escribir con mayúsculas, con minúsculas o de alguna combinación entre ellas.

Enteros

```
1234 //número decimal
-123 //un número negativo
0123 //número octal (83 decimal)
0x12 //número hexadecimal (18 decimal)
```

De punto flotante

Los números en punto flotante se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
1.234
1.2e3 //Notación cientifica
```

Cadenas

- Una cadena es una serie de caracteres. No existe un límite en el tamaño de una cadena.
- Pueden especificarse de varias maneras, las principales son:

‘Cadena entre comillas simples’

“Cadena entre comillas dobles”

Cadenas

- Cuando una cadena está encerrada en comillas simples, no se interpretará nada más que una comilla simple escapada y una diagonal invertida escapada.

  \’ = comilla simple escapada

  \\ = diagonal invertida escapada

- Cuando una cadena está encerrada en comillas dobles, se interpretarán posibles expresiones y variables contenidas en la cadena.

  \n = nueva línea

  \t = tabulador horizontal

  \” = comillas dobles

NULL

- NULL sirve para cuando no se tiene un valor asignado.
- El único valor para este tipo es:

NULL

Variables

- Una variable es un contenedor de información, en el que podemos almacenar números enteros, flotantes, cadenas, arreglos, etc.
- El contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.
- No es necesario declarar una variable antes de usarla.
- No tienen tipos definidos, es decir que una misma variable puede contener un número y luego puede contener una cadena de caracteres.

Declaración de variables

- En PHP todas las variables comienzan con el símbolo de dólar $
- PHP es sensible a mayúsculas y minúsculas.
- **$estadoCuenta** y **$EstadoCuenta** son dos variables distintas.
- Un nombre válido tiene que empezar con una letra o un guión bajo, seguido de cualquier número de letras, números y guiones bajos.

Cadenas con variables

- Comillas simples NO interpretarán las variables.
- Comillas dobles SI interpretan las variables.

```php
<?php
$x = 3;
echo 'ejemplo $x <br>';
echo "ejemplo $x <br>";

$one = 1;
$two =2;
$three = 3;
$four = 4;
$five =5;
$arr = array(5,6);

/* Las comillas dobles hacen mas legibles las concatenaciones */
$testDoble = " $one $two $three $four $five <br>";
echo $testDoble;

/* Las comillas simples.. no tanto */
$testSimple = ' ' . $one . ' ' . $two . ' ' . $three . $four . ' ' . $five . ' ';
echo $testSimple;
?>
```

ejemplo $x
ejemplo 3
1 2 3 4 5
1 2 34 5

Funciones

- Las funciones son una secuencia de instrucciones que cumplen con un objetivo en común, son útiles cuando se quiere hacer lo mismo varias veces y sólo cambian algunos detalles, entonces se utiliza una función para optimizar el código y hacerlo mas legible para otros programadores.

Sintaxis Básica

Nombre de la función

Lista de parámetros (separados por comas)

```
function nombre(param1, param2, ...)
{
        instrucción1;
        instrucción2;
        .........

        return valor_de_retorno;
}
```

Código de la función

Valor de regreso (opcional)

Ejemplo de función

```
function suma($a,$b){
  echo $a+$b;
}

suma (3,8);
```

Ejemplo de función

```php
<?
function promedio($num1, $num2)
{
    $promedio = ($num1 + $num2)/2;
    return $promedio;
}
```

Definición de la función

```php
echo promedio(4,6),"<br>";

$var = promedio($a,$b);

?>
```

Llamadas a la función.

Práctica :: Función pendiente

- En el archivo **pendiente.php**, crea una función que reciba como argumentos los puntos (x y y de inicio y fin) correspondientes a los puntos de una línea.
- La función debe calcular y regresar la pendiente de la línea representada por dichos puntos.
- Incluye una llamada a esa función para probarla.

Funciones para el manejo de variables

- PHP contiene varias funciones útiles para manejar variables, como:

isset()
empty()
print_r()
var_dump()
is_numeric()
is_bool()
is_null()
is_string()

- Consultar más funciones en:

http://www.php.net/manual/en/ref.var.php

Operadores

- Los operadores son componente esencial de cualquier lenguaje de programación. Con ellos podemos asignar, unir, cambiar o comparar valores de datos, cambiar el flujo del programa, etc.
- Los operadores son símbolos que representan operaciones sobre un valor. A continuación se listan los más importantes:

1) **Aritméticos**
2) **De incremento/decremento**
3) **De cadena**
4) **De asignación**
5) **De comparación**
6) **Lógicos**

Aritméticos

Nombre	Ejemplo	Resultado
Adición	$a + $b	Suma de $a y $b.
Substracción	$a - $b	Diferencia entre $a y $b.
Multiplicación	$a * $b	Producto de $a y $b.
División	$a / $b	Cociente de $a entre $b.
Módulo	$a % $b	Sobrante de $a dividido entre $b.

Práctica :: Operadores aritméticos

- En el archivo **operadores_aritmeticos.php**, crear las siguientes variables:
  $x=10;
  $y=7;
- Utilizando operadores, escribe el código para imprimir lo siguiente:

```
10 + 7 = 17
10 - 7 = 3
10 * 7 = 70
10 / 7 = 1.4285714285714
10 % 7 = 3
```

- Los números solo deben usarse en la asignación de variables. Si lo requieres puedes usar otra variable.

Incremento/Decremento

Nombre	Ejemplo	Resultado
Preincremento	++$a	Incrementa $a en uno y después devuelve $a.
Postincremento	$a++	Devuelve $a y después incrementa $a en uno.
Predecremento	--$a	Decrementa $a en uno y después devuelve $a.
Postdecremento	$a--	Devuelve $a y después decrementa $a en uno.

De asignación

Nombre	Ejemplo	Resultado
Asignación	$a = 3	$a se define como 3
	Combinados	
Asignación-Suma	$a += 3	Equivalente a $a = $a + 3
Asignación-Resta	$a -= 3	Equivalente a $a = $a - 3
Asignación-Multiplicación	$a *= 3	Equivalente a $a = $a - 3
Asignación-División	$a /= 3	Equivalente a $a = $a / 3
Asignación-Concatenación	$a .= ‘Mundo’	Equivalente a $a = $a . ‘Mundo’

Práctica :: Operadores de asignación

- Crear el script **operadores_asignacion.php**
- Utilizando una sola variable, y solamente operadores aritméticos de asignación e incremento/decremento, escribe el código para imprimir lo siguiente:

```
El valor ahora es 8.
Suma 2. El valor ahora es 10.
Resta 4. El valor ahora es 6.
Multiplica por 5. El valor ahora es 30.
Divide entre 3. El valor ahora es 10.
Incrementa el valor en 1. El valor ahora es 11.
Decrementa el valor en 1. El valor ahora es 10.
```

De cadenas

Nombre	Ejemplo	Resultado
Concatenación	$a . $b	Concatena (junta) $a y $b

De comparación

- Los operadores de comparación, como su nombre indica, permiten comparar dos valores. Su resultado es TRUE o FALSE.

Nombre	Ejemplo	Resultado
Igualdad	$a == $b	Cierto si $a es igual a $b.
Identidad	$a === $b	Cierto si $a es igual a $b y si son del mismo tipo.
Desigualdad	$a != $b	Cierto si $a no es igual a $b.
Menor que	$a < $b	Cierto si $a es estrictamente menor que $b.
Mayor que	$a > $b	Cierto si $a es estrictamente mayor que $b.
Menor o igual que	$a <= $b	Cierto si $a es mayor o igual que $b.
Mayor o igual que	$a >= $b	Cierto si $a es menor o igual que $b.

Lógicos

- Los operadores lógicos son utilizados para evaluar una expresión, el resultado depende si se cumple o no la condición. Su resultado es un valor booleano TRUE o FALSE.

Nombre	Ejemplo	Resultado
AND (y)	$a && $b	TRUE si tanto $a como $b son ciertos.
Or (o inclusivo)	$a || $b	TRUE si alguno $a o $b son ciertos.
Xor (o exclusivo)	$a xor $b	TRUE si $a es cierto o $b es cierto, pero no ambos a la vez.
Not (no)	! $a	TRUE si $a no es cierto.

Valores Falsos

- Los valores que se consideran como Falsos son:
  - El valor booleano FALSE
  - El entero 0 (cero)
  - El flotante 0 (cero)
  - Una cadena vacía, o la cadena “0”
  - Un arreglo con cero elementos
  - Un objeto con cero variables miembro
  - El tipo especial NULL

Arreglos

- Un arreglo es un contenedor de varios valores, es decir, es un conjunto de valores cada uno de los cuales está identificado por alguna clave. De esta manera se tienen agrupados esos valores, pero con la cualidad de poder identificar cada uno y realizar operaciones sobre el conjunto entero.
- Se trata de un mapa ordenado, donde cada valor está asociado a una llave.
- Se puede implementar como una lista, una pila, una cola, un diccionario, una tabla asociativa, entre otras estructuras de datos.
- Los valores de un arreglo pueden ser otros arreglos, por lo que también es posible crear árboles y arreglos multidimensionales.

Arreglos

- La llave puede ser únicamente un número entero o una cadena.
- El valor puede ser de cualquier tipo.
- Un arreglo puede crearse de 2 maneras:
  - Usando la construcción del lenguaje `array()`

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

  - Usando la sintaxis de corchetes [ ]

Estructuras de Control

- La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Condicionales:	**Iterativas:**
if	**while**
if..else	**do-while**
if..elseif..else	**for**
switch	**foreach**

if...elseif...else

```
if ($i == 0)
{
   print “i es igual a 0”;
}
elseif ($i == 1)
{
    print “i es igual a 1”;
}
else
{
    print “i es igual a 2”;
}
```

switch

```
switch ($i)
{
   case 0: print “i es igual a 0”;
           break;
   case 1: print “i es igual a 1”;
           break;
   case 2: print “i es igual a 2”;
           break;
   default: print “i no es 0 ni 1 ni 2”;
}
```

while

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

for

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

foreach

```php
//Se extrae solo el valor de cada elemento del arreglo
foreach ($arr as $value)
{
	echo "Value: $value<br>\n";
}

//Se extrae la llave y el valor de cada elemento del arreglo

foreach ($arr as $key => $value)
{
	echo "Key: $key; Valor: $value<br>\n";
}
```

Break

- break termina la ejecución de la estructura actual for, foreach, while, do-while o switch.
- break acepta un argumento numérico opcional el cual indica de cuantas estructuras anidadas encerradas se debe salir.

Continue

- continue se utiliza dentro de las estructuras de bucle para saltarse el resto de la actual iteración del bucle y continuar la ejecución en la evaluación de la condición y entonces el comienzo de la siguiente iteración.
- continue acepta un argumento numérico opcional el cual indica hasta el final de cuantos niveles de bucles cerrados se debe saltar.

Operador Ternario

- Otro operador condicional es el operador "?:" (o ternario), que evalúa una expresión booleana y regresa uno de dos valores.

- Donde expr1 se evalúa de manera booleana; si es TRUE se toma el valor de expr2, si es FALSE se toma el valor de expr3

```
(expr1) ? (expr2) : (expr3);
```

```
$a==0 ? “cero” : “no cero”;
```

Desarrollo de software con PhP

Jonathan Leonardo Silva Blasio
bls.jonathan.proyectos@gmail.com

Arreglos

- Un arreglo es un contenedor de varios valores, es decir, es un conjunto de valores cada uno de los cuales está identificado por alguna clave. De esta manera se tienen agrupados esos valores, pero con la cualidad de poder identificar cada uno y realizar operaciones sobre el conjunto entero.
- Se trata de un mapa ordenado, donde cada valor está asociado a una llave.
- Se puede implementar como una lista, una pila, una cola, un diccionario, una tabla asociativa, entre otras estructuras de datos.
- Los valores de un arreglo pueden ser otros arreglos, por lo que también es posible crear árboles y arreglos multidimensionales.

Arreglos

- La llave puede ser únicamente un número entero o una cadena.
- El valor puede ser de cualquier tipo.
- Un arreglo puede crearse de 2 maneras:
  - Usando la construcción del lenguaje `array()`

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

  - Usando la sintaxis de corchetes [ ]

    [‘platano’, ‘fresa’, ‘manzana’, ‘pera’, ‘sandía’]

Constantes

- Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script. Una constante es sensible a mayúsculas.
- Por convención, los identificadores de constantes suelen declararse en mayúsculas
- Se puede acceder a constantes desde cualquier sitio del script sin importar desde donde.
- Una vez definida, no puede ser modificada ni eliminada.
- Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

Constantes

- Las constantes no son precedidas por un símbolo de dólar ($).
- Las constantes solo pueden ser definidas usando la función define() , nunca por simple asignación.

```
define(“NOMBRE_CONSTANTE",  “valor_constante”);

define("CONSTANTE_SALUDO", “Hola Mundo!”);

print CONSTANTE; // imprime “Hola Mundo!”
```

4. HERRAMIENTAS ÚTILES

INCLUSION DE ARCHIVOS

Inclusión de archivos

- A medida que aumenta el número de páginas y funciones generadas, se hace necesario algún mecanismo para que permita utilizar el código escrito en otro archivo.

```
require(nombre_archivo)
require_once(nombre_archivo)
include(nombre_archivo)
include_once(nombre_archivo)
```

- Estas instrucciones permiten insertar y evaluar un archivo en el programa que actualmente se está cargando.

Inclusión de archivos

- En caso de error **require** genera un error fatal que interrumpe la ejecución.
- En cambio **include** genera solamente un error de advertencia (warning) que permite seguir procesando el archivo.
- **require_once** e **include_once** validan que un archivo determinado se incluya únicamente una vez.

```
require('usuarios.php')
require_once('/funciones.php')
include('./constantes.php')
include_once('../tmp/baseDatos.php')
```

Inclusión de archivos

- Cuando un fichero es incluido, el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final.
- Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

USO DE BASE DE DATOS

Usar una base de datos

1. Conectarse al manejador.
2. Conectarse a la base de datos.
3. Ejecutar sentencias SQL para obtener, insertar, actualizar y eliminar datos.
4. Ejecutar sentencias SQL para modificar la estructura.
5. Desconectarse de la base de datos.

Conexión y desconexión

```php
<?php
$link = pg_connect("host=localhost port=5432 dbname=registro user=postgres password=hola");

if (!$link) {
    die('Error de Conexión' );
}

//Usar la base (inserts, updates, deletes, selects)

pg_close ($link);
?>
```

Ejecutar sentencias DML
(SELECT, INSERT, UPDATE, DELETE)

- Regresa **false** si ocurre un error
- Regresa **un result set** en caso de exito

```
$resultSet = pg_query ($link , $query);
```

Ejecutar sentencias DML (select)

- Se utiliza también **pg_query**, pero en caso de éxito regresa un **resultset**, el cual contiene los datos que se seleccionaron.
- Con el **resultset** es posible obtener cada fila del resultado de diversas maneras:

pg_fetch_assoc ($result) ← **Como un array asociativo**
pg_fetch_row ($result) ← **Como un array enumerado**

Función de re direccionamiento

- Utilizamos la función header para redireccionar a otra página al navegador
- Esta función corta el flujo de nuestro script de php

```
header("Location: usuario.php");
header("Location: www.google.com");
```

VARIABLES SUPERGLOBALES

PROCESAMIENTO DE FORMULARIOS

MANEJO DE SESIONES

Supervariables

- PHP ofrece un gran número de variables predefinidas a todos los scripts. Las variables representan todo desde variables externas a variables incorporadas de entorno, últimos mensajes de error hasta las últimas cabeceras recuperadas.

`$_GET['variable']` - Variables proporcionadas vía HTTP GET.

`$_POST['variable']` - Variables proporcionadas vía HTTP POST.

`$_COOKIE['variable']` - Variables proporcionadas vía HTTP cookies.

Supervariables

$_REQUEST['variable'] - Variables proporcionadas via GET, POST, o COOKIE.

$GLOBALS['variable'] - Array con todas las variables de alcance global.

$_SERVER['variable'] - Variables vía servidor o relacionadas con el entorno de ejecución.

$_FILES['variable'] - Variables vía upload archivos, es un arreglo de arreglos.

$_ENV['variable'] - Variables del entorno

$_SESSION['variable'] - Variables registradas en una sesión

Recibir las variables

- Un usuario puede hacerle llegar información a PHP a través de los métodos POST y GET.
- PHP guarda las variables que se envían a través de estos métodos en los arreglos asociativos

$_GET y **$_POST**

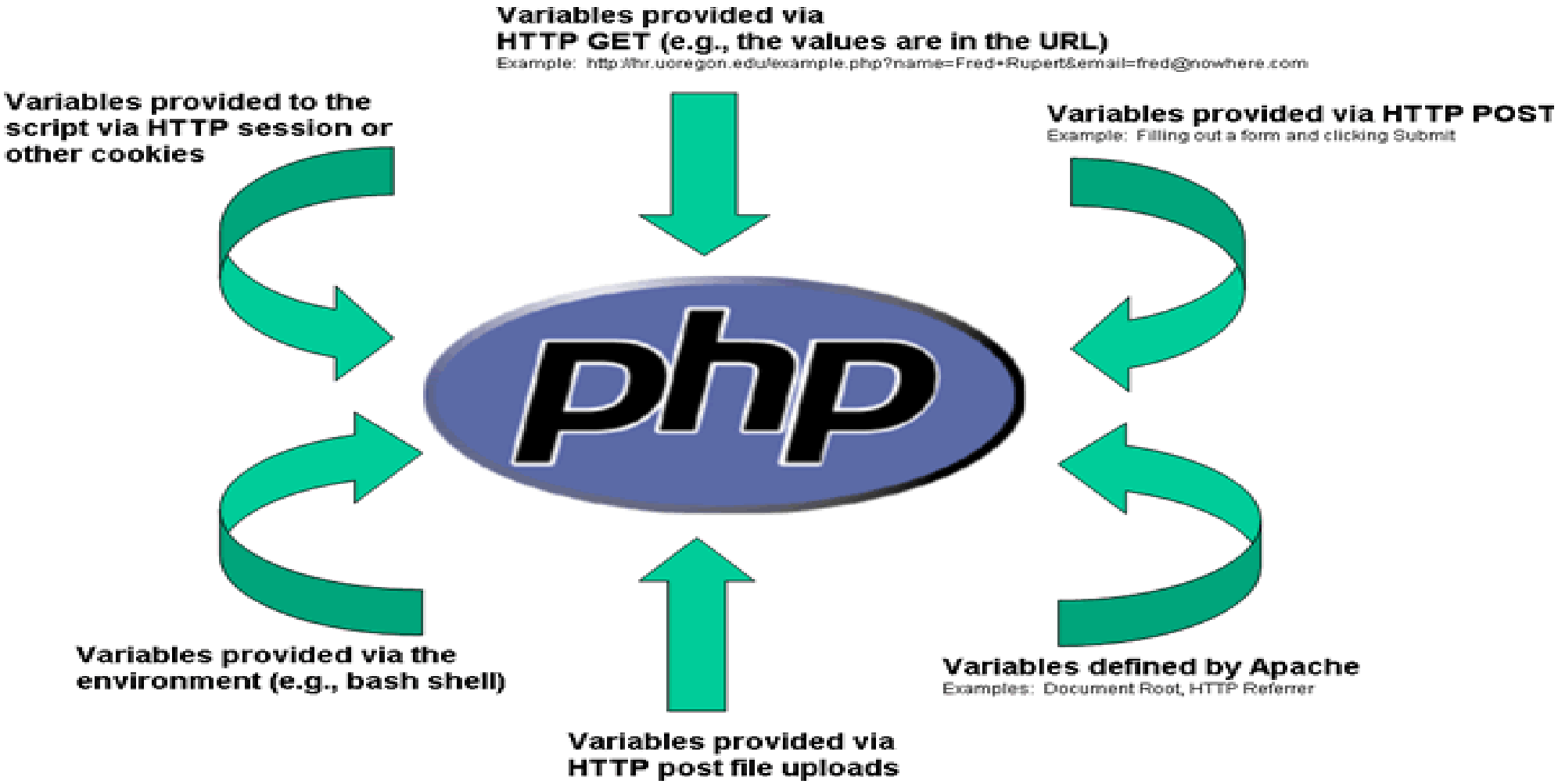
- Se tiene acceso a las variables ocupando el nombre del input como clave del arreglo:

```
$_GET[“var1”]
$_POST[“var1”]
```

Formularios HTML
(GET y POST)

- Cuando se envía un formulario a un script PHP, las variables de dicho formulario quedan automáticamente disponibles en el script gracias a PHP.

Recibir las variables



ENVÍO DE ARCHIVOS

Envío de archivos

- PHP es capaz de recibir envíos de archivos de cualquier navegador. Ésta característica permite que los usuarios envíen archivos de texto y binarios.
- Mediante la autenticación y funciones de manejo de archivos de PHP, es posible un control total de quién puede enviar archivos y que se hace con éstos una vez recibidos.
- Los archivos serán almacenados en el directorio temporal por defecto del servidor.

Envío de archivos

- Lo primero que se debe hacer es crear un formulario HTML, que permita al usuario abrir una ventana de diálogo para seleccionar el archivo que se desea enviar:

```
<FORM method="POST" ENCTYPE="multipart/form-data">
<INPUT type=“hidden” name=“MAX_FILE_SIZE” VALUE=“2048”>
<INPUT type=“file” name="nombre_campo">
<INPUT type=submit value="Enviar">
</FORM>
```

- Es importante que no se olvide incluir el atributo ENCTYPE="multipart/form-data" de lo contrario, no se podrá enviar el archivo.

Envío de archivos

- La información de los archivos enviados se almacena en la superglobal $_FILES, que es un arreglo.
- A continuación se describe el contenido de $_FILES:

$_FILES[‘nombre_campo’]['name'] - El nombre original del archivo en la máquina cliente.

$_FILES['nombre_campo']['type'] - El tipo mime del archivo (si el navegador lo proporciona). Un ejemplo podría ser "image/gif".

$_FILES['nombre_campo']['size'] - El tamaño en bytes del archivo recibido.

$_FILES['nombre_campo']['tmp_name'] - El nombre del archivo temporal que se utiliza para almacenar en el servidor el archivo recibido.

$_FILES['nombre_campo']['error'] – El código de error o éxito.

Envío de archivos

- La función **move_uploaded_files()** transfiere la imagen del directorio temporal a un directorio de destino y la funcion **is_uploaded_file()** , indica si un archivo fue cargado a través de POST.

```
if (is_uploaded_file($_FILES['userfile']['tmp_name'])) {
   move_uploaded_file($_FILES['userfile']['tmp_name'],
      "/home/files");
} else {
   echo "El archivo ".$_FILES['userfile']['name'],
      "No fue enviado por POST";
}
```

SESIONES

Sesiones

- Básicamente una sesión es la secuencia de páginas que un usuario visita en un sitio web. Desde que entra en nuestro sitio, hasta que lo abandona.
- El soporte para sesiones en PHP consiste en una forma de preservar cierta información a través de accesos subsiguientes.
- PHP nos permite llevar el control de una sesión, almacenando información en el servidor

Sesiones

- Para hacer uso de las sesiones debemos indicárselo al intérprete con **session_start()** al inicio de cada script.
- Para guardar información de la sesión se utiliza la variable superglobal **$_SESSION**
- Para destruir una variable se utiliza

  unset ($_SESSION['nombre_variable']);
- Para destruir la sesión completa se utiliza

  session_destroy()

Ejercicio: Autenticación