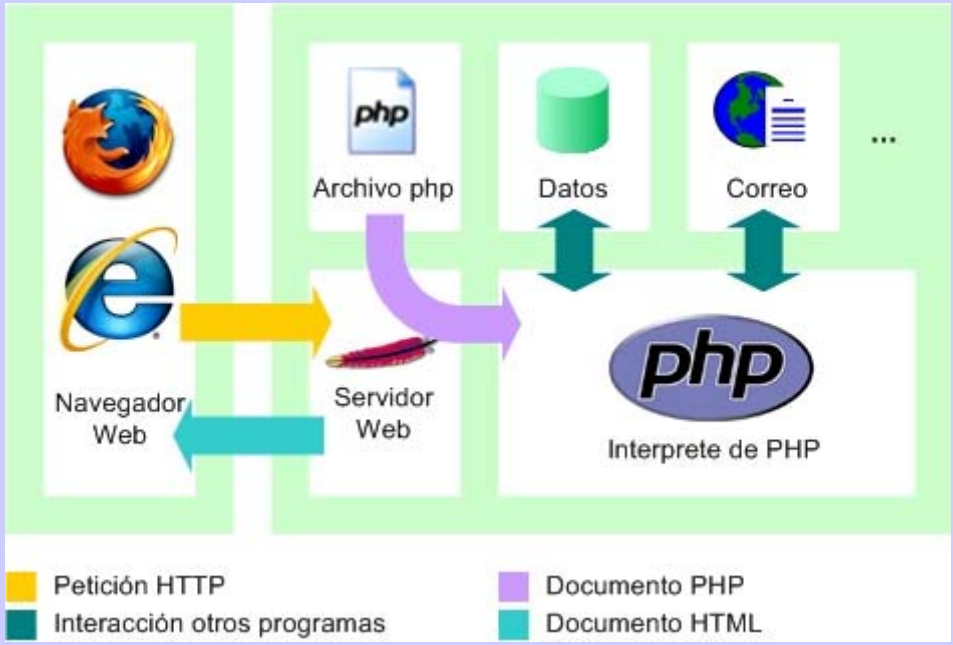
- PHP
  - Personal Hypertext Preprocessor
  - PHP: Hypertext Preprocessor
  - Lenguaje de alto nivel, de código abierto, interpretado
  - Código embebido en páginas HTML y ejecutado en el servidor


php
Archivo php
Datos
Correo
...
Navegador Web
Servidor Web
php
Interprete de PHP
Petición HTTP
Interacción otros programas
Documento PHP
Documento HTML

- Primer ejemplo

```
<html>
<head>
  <title>Primer ejemplo</title>
</head>
<body>
  <?php
    echo "<p>Hola mundo</p>";
  ?>
</body>
</html>
```

- Primer ejemplo, resultado

```
<html>
<head>
  <title>Primer ejemplo</title>
</head>
<body>
  <p>Hola mundo</p>
</body>
</html>
```

- Segundo ejemplo (formulario en un html)

```
<form action="accion.php" method="POST">
  Nombre: <input type="text" name="nombre" />
  Edad: <input type="text" name="edad" />
  <input type="submit">
</form>
```

- Segundo ejemplo (accion.php)

```
<html>
<head>
  <title>Segundo ejemplo</title>
</head>
<body>
  Hola <?php echo $_POST["nombre"]; ?>
  Tiene <?php echo $_POST["edad"]; ?> a&ntilde;os
</body>
</html>
```

- Tercer ejemplo (accion.php)

```
<html>
<head>
  <title>Tercer ejemplo</title>
</head>
<body>
  <?php
    if ($_POST["edad"] > 18) {
      echo "Hola " . $_POST["nombre"];
      echo "Tiene " . $_POST["edad"] . " a&ntilde;os";
    }
    else {
      echo "Menor de edad";
    }
  ?>
</body>
</html>
```

- Cuarto ejemplo (accion.php)

```
<html>
<head><title>Cuarto ejemplo</title></head>
<body>
  <?php
    if ($_POST["edad"] > 18) {
  ?>
      Hola <?php echo $_POST["nombre"]; ?>
      Tiene <?php echo $_POST["edad"]; ?> a&ntilde;os
  <?php
    }
    else {
  ?>
      Menor de edad
  <?php
    }
  ?>
</body>
</html>
```

- PHP – Sintaxis
  - Comentarios

```
<?php
  echo "Texto"; // Comentario de una línea
  /* Comentario de
     varias líneas */
?>
```

  - Secuencia: instrucciones separadas por “;”

```
<?php
    echo "Texto";
?>
<?php echo "Texto" ?><!-- fin de bloque, ";" implícito -->
```

- Tipos
  - No se requiere definición de tipos en la declaración de variables
  - El tipo de una variable es determinado por el contexto en el que la variable es usada.
    - Si se asigna un valor de cadena a la variable $var , $var se convierte en una cadena.
    - Si luego se asigna un valor entero a $var , ésta se convierte en entera.

- Tipos
  - Boolean
    - Valores: `true` `false` (case insensitive)
    - En realidad: false es 0 y true cualquier otro valor
    - Entonces:

```
false      // false
""         // false
0          // false
1          // true
-2         // true
"foo"      // true
2.3e5      // true
"false"    // true
```

- Tipos
  - Enteros
    - `$a = 1234; // número decimal`
    - `$a = -123; // un número negativo`
    - `$a = 0123; // número octal (83)`
    - `$a = 0x1A; // número hexadecimal (26)`
  - Flotantes
    - `$a = 1.234;`
    - `$b = 1.2e3;`
    - `$c = 7E-10;`

- Tipos
  - Strings
    - Comillas dobles

```
echo "Cadena de caracteres";
echo "Cadena de caracteres \n con salto de línea";
echo "Arnold dijo una vez: \"I'll be back\"";
echo "Ha eliminado C:\\*.*?";
echo "Las variables se $expanden";
```

    - Comillas simples

```
echo 'Cadena de caracteres';
echo 'Cadena de caracteres
      con salto de línea';
echo 'Arnold dijo una vez: "I\'ll be back"';
echo 'Ha eliminado C:\\*.*?';
echo 'Ha eliminado C:\*.*?';
echo 'Esto no va a expandirse: \n una nueva línea';
echo 'Las variables no se $expanden';
```

- Tipos
  - Arreglos
    - `$vector = array(clave => valor, ...);`
    - clave → string o integer
    - valor → cualquier tipo
    - `$vector = array("foo" => "bar", 12 => true);`

    - `echo $vector["foo"]; // bar`
    - `echo $vector[12]; // 1`

- Tipos
  - Arreglos
    - Omisión de clave

```
array(5 => 43, 32, 56, "b" => 12); // es lo mismo que
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
```

    - Agregar elemento

```
$vector = array(5 => 1, 12 => 2);
$vector[13] = 42; // Esto agrega un nuevo elemento
$vector[] = 50; // Esto también. Si no existe, lo crea
```

    - Eliminar par clave/valor

```
unset($vector[5]);
```

    - Eliminar vector

```
unset($vector);
```

- Tipos
  - Arreglos. Ejemplo

```
$a = array(1, 2, 3, 4, 5); // Crear un array
print_r($a);
// Elimina cada ítem pero no el vector
foreach ($a as $clave => $valor) {
    unset($a[$clave]);
}
print_r($a);
$a[] = 6; // Agrega un ítem (la nueva clave es 5)
print_r($a);
$a = array_values($a); // Re-indexa
$a[] = 7; // Agrega un ítem
print_r($a);
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
)
```

- Tipos
  - Arreglo. Ejemplo

```
$cambios = array(        10, // clave = 0
                  5  =>  6,
                  3  =>  7,
                 'a' =>  4,
                         11, // clave = 6 (índice entero máximo era 5)
                 '8' =>  2, // clave = 8 (¡entero!)
                '02' => 77, // clave = '02
                   0 => 12  // el valor 10 reemplazado por 12
           );
```

  - Arreglo vacío. Ejemplo

```
$vacio = array();
```

- Tipos
  - Arreglos. Referenciación (&)

```
$a1 = array(2, 3);
$a2 = $a1;
$a2[] = 4; // $a2 cambia,
           // $a1 sigue siendo array(2, 3)

$a3 = &$a1;
$a3[] = 4; // $a1 y $a3 son iguales
```

- Tipos
  - Matrices

```
$matriz = array("unamatriz" => array(  6 => 5,
                                      13 => 9,
                                     "a" => 42));

echo $matriz["unamatriz"][6];    // 5
echo $matriz["unamatriz"][13];   // 9
echo $matriz["unamatriz"]["a"];  // 42
```

- Tipos
  - Matrices

```
$frutas = array("frutas"  => array("a" => "naranja",
                                   "b" => "banana",
                                   "c" => "manzana"),
                "numeros" => array(1, 2, 3, 4, 5, 6),
                "hoyos"   => array(      "primero",
                                    5 => "segundo",
                                         "tercero")
          );

// Algunos ejemplos
echo $frutas["hoyos"][5];    // imprime "segundo"
echo $frutas["frutas"]["a"]; // imprime "naranja"
unset($frutas["hoyos"][0]);  // elimina "primero"

// Qué hace?
$jugos["manzana"]["verde"] = "bien";
```

- Tipos
  - Null
    - Indica que una variable NO tiene valor, porque
      - se ha asignado la constante NULL a la variable
      - no ha sido definida con valor alguno
      - ha sido eliminada con unset()

```
$var = NULL;
```

  - Función `is_null()`

```
if (is_null($var)) { ... }
```

- Tipos
  - Casting (conversión de tipos)
    - Implícito

```
$var = "0"; // $var es un string (ASCII 48)
$var += 2;  // ahora es ahora un entero (2)
$var = $var + 1.3; // ahora es un flotante (3.3)
$var = 5 + "10 Cerditos"; // $var es entero (15)
$var = 5 + "10 Cerdos";   // $var es entero (15)
```

    - Explícito
      - Se antepone el nombre del tipo entre paréntesis

```
$var = "10";                // $var es un string
$var = (boolean) $var;      // $var es un boolean
$var = (integer) $var;      // $var es un entero
```

- Variables
  - Signo $ seguido del nombre de la variable
  - Case sensitive
  - Empieza con letra o "_", seguido de letras, número o "_"
  - Asignación
    - Por valor
      -
```
$a = 4 + 5
$a = $b
```
    - Por referencia
      -
```
$a = &$b
```

- Variables
  - Variables predefinidas
    - $GLOBALS
      - Matriz de referencia a variables “alcanzables”. Las claves son los nombres de las variables
    - $_GET
      - Parámetros enviados al script por el método GET
    - $_POST
      - Parámetros enviados al script por el método POST
    - $_COOKIE
    - $_FILES
      - Matriz asociativa de elementos cargados al script actual por el método POST
    - $_REQUEST
      - Matriz asociativa con los contenidos de $_GET, $_POST, y $_COOKIE
    - $_SESSION

- Variables
  - Alcance

```
<?php
  $a = 1;    // alcance global
  function Test() {
    echo $a; // alcance local
  }
  Test();    // no produce salida
?>
```

- Variables
  - Alcance

```
<?php
  $a = 1;
  $b = 2;
  function Sum() {
    global $a, $b; /* declara que $a y $b
                      hacen referencia a
                      las globales      */
    $b = $a + $b;
  }

  Sum();
  echo $b; // Salida 3
?>
```

- Variables
  - Alcance

```
<?php
  $a = 1;
  $b = 2;
  function Sum() {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
  }

  Sum();
  echo $b; // Salida 3
?>
```

- Variables estáticas

```
function Test () {
  $a = 0;
  echo $a;
  $a++
}

function Test() {
  static $a = 0;
  echo $a;
  $a++;
}
```

- Variables estáticas en recursión

```
function Test() {
  static $count = 0;
  $count++;
  echo $count;
  if ($count < 10) {
    Test();
  }
  $count--;
}
```

- Variables variables
  - El valor de una variable se usa como nombre de variable

```
$a = "hello";
$$a = "world"; // es lo mismo que $hello = "world";
echo "$a ${$a}";  // hello world
echo "$a $hello"; // hello world
```

- Constantes
  - Se define por medio de la función `define()`
  - No pueden ser modificadas ni eliminadas
  - Sólo valores escalares (boolean, integer, float y string)

```
define("CONSTANTE", "Hola Mundo");
```

  - Para saber si una constante está definida:

```
if (defined("CONSTANTE")) {
    echo CONSTANTE;
}
```

- Operadores
  - Aritméticos

`-$a`	Negación	El opuesto de $a
`$a + $b`	Adición	Suma de $a y $b
`$a - $b`	Substracción	Diferencia entre $a y $b
`$a * $b`	Multiplicación	Producto de $a y $b
`$a / $b`	División	Cociente de $a y $b
`$a % $b`	Módulo	Resto de $a dividido por $b

- Operadores
  - Incremento y decremento

`++$a`	Pre-incremento	Incrementa $a en uno, y luego devuelve $a
`$a++`	Post-incremento	Devuelve $a, y luego incrementa $a en uno
`--$a`	Pre-decremento	Decrementa $a en uno, luego devuelve $a
`$a--`	Post-decremento	Devuelve $a, luego decrementa $a en uno

  - De Strings

`$a.$b`	Concatenación	Concatenar $a y $b

- Operadores
  - Bit a bit

`$a & $b` Y

Son activados los bits que están activos tanto en $a como en $b

`$a | $b` O

Son activados los bits que están activos ya sea en $a o en $b

`$a ^ $b` O exclusivo (Xor)

Son activados los bits que estén activos en $a o $b, pero no en ambos

`~ $a` No

Son activados los bits que estén desactivos en $a y vice-versa

`$a << $b` Desplazamiento a izquierda

Desplaza los bits de $a, $b pasos a la izquierda (cada paso es equivalente a "multiplicar por dos")

`$a >> $b` Desplazamiento a derecha

Desplaza los bits de $a, $b pasos a la derecha (cada paso es equivalente a "dividir por dos")

- Operadores
  - Comparación

`$a == $b`	Igual	TRUE si $a es igual a $b
`$a === $b`	Idéntico	TRUE si $a es igual a $b, y son del mismo tipo
`$a != $b`	Diferente	TRUE si $a no es igual a $b
`$a !== $b`	No idénticos	TRUE si $a no es igual a $b, o si no son del mismo tipo
`$a < $b`	Menor que	TRUE si $a es estrictamente menor que $b
`$a > $b`	Mayor que	TRUE si $a es estrictamente mayor que $b
`$a <= $b`	Menor o igual que	TRUE si $a es menor o igual que $b
`$a >= $b`	Mayor o igual que	TRUE si $a es mayor o igual que $b

- Operadores
  - De asignación

```
=          $a = $b
+=         $a = $a + $b
-=         $a = $a - $b
*=         $a = $a * $b
/=         $a = $a / $b
.=         $a = $a . $b
%=         $a = $a % $b
&=         $a = $a & $b
|=         $a = $a | $b
^=         $a = $a ^ $b
<<=        $a = $a << $b
>>=        $a = $a >> $b
```

- Operadores
  - Ternario
    - `(expr1) ? (expr2) : (expr3)`
    - `$max = $a > $max ? $a : $max`
    - Cuál es la salida de:

```
echo (true ? 'true' : false ? 't' : 'f');

echo ((true ? 'true' : false) ? 't' : 'f');
ó
echo (true ? 'true' : (false ? 't' : 'f'));
```

- Expresiones
  - `$a = 1;` es una “expresión”
    - Devuelve el valor asignado
  - Entonces se podría:
    - `$b = ($a = 1);`
  - O lo que es lo mismo:
    - `$b = $a = 1;`

- Expresiones
  - Algunos ejemplos

```
<?php
  function double($i) {
    return $i*2;
  }
  $b = $a = 5;         // asigna 5 a $a y $b
  $c = $a++;           // postincremento, $c es 5 y $a es 6
  $e = $d = ++$b;      // preincremento, $b, $d y $e son 6

  $f = double($d++);   // $f es 12 y $d es 7
  $g = double(++$e);   // $g es 14 y $e es 7
  $h = $g += 10;       // $g es 24 y $h es 24
?>
```

- Estructuras de control
  - Selección

```
if (condición) {
  sentencias if;
}


if (condición) {
  sentencias if;
}
else {
  sentencias else;
}
```

- Estructuras de control
  - Selección

```
if (condición if) {
  sentencias 1;
}
elsif (condición elsif 1) {
 sentencias elsif 1;
}
elsif (condición elsif 2) {
 sentencias elsif 2;
}
……
else {
 sentencias else;
}
```

- Estructuras de control
  - Selección

```
switch (expresión) {
  case valor1:
    sentencia11;
    ……;
    break;
  ……..
  case valorN:
    sentenciaN1;
    ……;
    break;
  default:
    sentencias;
}
```

- Estructuras de control
  - Iteración

```
while (expresión) {
  sentencias;
}

$i = 0;
while ($i <= 10) {
  echo ++$i;
}
```

- Estructuras de control
  - Iteración

```
do {
  sentencias;
} while (expresión);
```

```
$i = 0;
do {
  echo ++$i;
} while ($i < 10);
```

- Estructuras de control
  - Iteración

```
for (expresión1; expresión2; expresión3) {
  sentencias;
}
```

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

- Estructuras de control
  - Iteración

```
foreach (expresiónArray as $value) {
  sentencias;
}

foreach (expresiónArray as $key => $value) {
  sentencias;
}
```

- Estructuras de control
  - Iteración

```
$arr = array(1, 2, 3, 4, 5);
foreach ($arr as &$value) {
  echo $value;
}
foreach ($arr as $key => $value) {
    echo "$key - $value";
}

unset($key);
unset($value);
```

- Estructuras de control
  - Iteración

```
$a = array();
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "c";
$a[1][1] = "d";

foreach ($a as $v1) {
  foreach ($v1 as $v2) {
    echo "$v2\n";
  }
}
```

- Más sentencias
  - include()
    - Incluye y evalúa el archivo dado como parámetro

vars.php

```
<?php
$color = 'green';
$fruit = 'apple';
?>
```

test.php

```
<?php
echo "A $color $fruit"; // A
include 'vars.php';
echo "A $color $fruit"; // A green apple
?>
```

- Más sentencias
  - include()

vars.php

```
<?php
$color = 'green';
$fruit = 'apple';
?>
```

```
<?php
function foo() {
  global $color;
  include 'vars.php';
  echo "A $color $fruit";
}

foo();                          // A green apple
echo "A $color $fruit";         // A green
?>
```

- Más sentencias
  - require()
    - Es idéntico al include excepto que si falla produce un error fatal E_ERROR (detiene la ejecución del script), mientras que si el include falla sólo emite un warning (E_WARNING), el cual permite al script seguir ejecutando
  - require_once()
    - Idéntico al require() excepto que verifica si el archivo ya fue incluido, en cuyo caso no vuelve a incluirlo (requerirlo)
  - include_once()
    - Igual comportamiento que el include, excepto que si el archivo ya fue incluido, no vuelve a incluirlo

- Funciones definidas por el usuario

```
<?php
function func($param1, $param2) {
  echo "Ejemplo.\n";
  return $retval;
}

func(1, 'a');
echo func(1, 'a');
?>
```

- Funciones definidas por el usuario
  - Las funciones no requieren estar definidas antes de ser invocadas, excepto cuando están definidas condicionalmente

```
<?php
$makefoo = true;

bar();

if ($makefoo) {
  function foo() {
    echo "No existe hasta que se ejecute el cuerpo del if.\n";
  }
}

if ($makefoo) foo();

function bar() {
  echo "Existe desde el comienzo del programa.\n";
}
?>
```

- Funciones definidas por el usuario

```
<?php
function foo() {
  function bar() {
    echo "No existe hasta que foo() sea
          invocada.\n";
  }
}

foo();
bar();
?>
```

  - Todas las funciones tienen alcance global (aunque hayan sido definidas dentro de otra)

- Funciones definidas por el usuario – Recursión

```
<?php
function sumatoria($a) {
  if ($a > 0)
    return $a + sumatoria($a - 1);
  else
    return 0;
}
?>

<?php
function sumatoria($a) {
  return ($a > 0) ? ($a + sumatoria($a - 1)) : 0;
}
?>
```

- Pasaje de parámetros
  - Por defecto “por valor”

```
<?php
function print_string($string) {
  echo "Parámetro $string";
}
$str = 'This is a string';
print_string($str);
print_string('This is a string');
print_string($str . '.');
?>
```

- Pasaje de parámetros
  - Pero también “por referencia”

```
<?php
function add_some_extra(&$string) {
  $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // 'This is a string, and
                 something extra.'
?>
```

- Pasaje de parámetros
  - Valores de parámetros por default

```
<?php
function makecoffee($type = "cappuccino") {
  return "Making a cup of $type.\n";
}
echo makecoffee();
echo makecoffee(null);
echo makecoffee("espresso");
?>
```

- Pasaje de parámetros
  - Valores de parámetros por default
    - Los parámetros que tienen valores por default deben estar a la derecha de los que no tienen

```
<?php
function makeyogurt($type = "acidophilus",
  $flavour) {
   return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt("raspberry");
?>
```

- Pasaje de parámetros
  - Cantidad variable de parámetros
    - func_num_args
      - Retorna la cantidad de parámetros que recibe la función

```
$numargs = func_num_args();
echo "Number of arguments: $numargs\n";
```

    - func_get_arg
      - Dada la posiciòn de un parámetro (comenzando en cero), retorna el parámetro de dicha posiciòn

```
echo "Second argument is: " . func_get_arg(1);
```

    - func_get_args
      - Retorna un arreglo con la lista de los parámetros recibidos por la función

- Return
  - La sentencia return puede devolver valores de cualquier tipo
  - Causa la terminación de la ejecución de la función y devuelve el control a la línea que sigue a la invocación
  - Si es ejecutada desde un alcance global, será terminada la ejecución del script actual.
  - Si es ejecutada desde un archivo incluído (con include) o requerido (con require), se devuelve el control al archivo llamador. Si se trata de un include, el valor indicado por el return es devuelto por el include.

- Variables función

```
<?php
function foo() {
  echo "In foo()<br />\n";
}

function bar($arg = '') {
  echo "In bar(); argument was '$arg'.<br />\n";
}

function echoit($string) {
  echo $string;
}

$func = 'foo';
$func();        // This calls foo()

$func = 'bar';
$func('test');  // This calls bar()

$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

- Funciones anónimas

```
<?php
$greet = function($name) {
  printf("Hello %s\r\n", $name);
};

$greet('World');
$greet('PHP');
?>
```