Clase 02

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Clase 2

¡Te damos la bienvenida a la materia **Programación Web II**!

En esta clase vamos a ver los siguientes temas:

- Alcance o ámbito de las variables.
- Constructores del lenguaje echo y print.
- Tipos de datos.
- Strings.
- Funciones de strings.
- Números.
- Casteo.

Alcance o ámbito de las variables

El ámbito de una variable en PHP se refiere al contexto en el que una variable está definida y en el que es accesible.
El ámbito de una variable es muy similar a su alcance, y en algunos casos los términos son intercambiables.

El ámbito de una variable puede ser global o local, dependiendo de dónde se define. Las variables definidas fuera de cualquier función o método tienen un ámbito global, lo que significa que pueden ser accedidas y modificadas en cualquier parte del script.

Por otro lado, las variables definidas dentro de una función o método tienen un ámbito local, lo que significa que solo pueden ser accedidas y modificadas dentro de esa función o método.

Es importante tener en cuenta el ámbito de las variables al diseñar y escribir código en PHP, ya que puede afectar el comportamiento y la legibilidad del mismo.

Al trabajar con variables de ámbito global, es importante utilizarlas de manera responsable y evitar la sobreescritura accidental de valores. Por otro lado, al trabajar con variables de ámbito local, es importante tener en cuenta que no se comparten entre diferentes funciones o métodos y que pueden tener diferentes valores en diferentes partes del script.

Alcance global:
Una variable con alcance global puede ser accedida y modificada en cualquier parte del script, incluyendo dentro de funciones y métodos. Para hacer que una variable tenga alcance global, debe ser declarada fuera de cualquier función o método con la palabra clave "global".

Ejemplo de una variable con alcance global

```php
<?php

   $numero = 10;

   function multiplicarPorDos(){
   global $numero;
   $numero *= 2;
   }

   multiplicarPorDos();
   echo $numero; // Salida: 20

?>
```

Alcance local:
Una variable con alcance local solo puede ser accedida y modificada dentro de la función o método en la que fue declarada. Por defecto, todas las variables son de alcance local dentro de una función o método.

Ejemplo de una variable con alcance local

```php
<?php

   function mostrarNumero(){
       $numero = 5;
       echo $numero; // 5
   }

   mostrarNumero();
   echo $numero; // Salida: Notice: Undefined variable

?>
```

Constructores del lenguaje echo y print

Las funciones echo y print son dos maneras de imprimir o mostrar información en una página web usando PHP.

"echo" es una función más rápida y permite imprimir múltiples argumentos separados por comas.

Ejemplo de uso del constructor echo:

```php
<?php

   echo "Hola";
   echo " mundo";

   // Salida: Hola mundo

?>
```

Ejemplo de uso del constructor echo con múltiples argumentos:

```php
<?php

   echo "Hola,", " mundo";

  // Salida: Esto imprimirá: "Hola, mundo".

?>
```

Es importante destacar que los argumentos deben separarse con una coma (,) y no con un punto y coma (;).

“print” es una función más lenta y solo permite imprimir un argumento. Además, print devuelve 1 como resultado, lo que lo hace útil para usar en expresiones.

Ejemplo de uso del constructor print:

```php
<?php

   print "Hola mundo"; // Salida: Hola mundo

?>
```

En general, se recomienda usar echo ya que es más rápido y flexible que print.

Tipos de datos

PHP soporta varios tipos de datos, incluyendo:

- **Booleanos:** valores lógicos verdadero (true) o falso (false).
- **Números enteros:** valores numéricos sin parte decimal.
- **Números de punto flotante:** valores numéricos con parte decimal.
- **Cadenas de caracteres:** secuencias de uno o más caracteres entre comillas simples o dobles.
- **Arreglos:** estructuras de datos que contienen una colección de valores, cada uno accesible a través de un índice o clave.
- **Objetos:** estructuras de datos que representan entidades con propiedades y métodos.
- **Recursos:** valores especiales que representan una conexión externa, como un archivo o una conexión a una base de datos.
- **Null:** un valor especial que representa la ausencia de un valor.

Ejemplo de tipo de dato booleano

```php
<?php

   $es_verdadero = true;
   $es_falso = false;

?>
```

Ejemplo de tipo de dato números enteros

```php
<?php

   $entero_positivo = 42;
   $entero_negativo = -42;

?>
```

Ejemplo de tipo de dato números de punto flotante

```php
<?php

   $flotante_positivo = 3.14;
   $flotante_negativo = -3.14;

?>
```

Ejemplo de tipo de dato cadena de caracteres

```php
<?php

   $cadena_simple = 'Hola mundo';
   $cadena_doble = "Hola mundo";

?>
```

Ejemplo de tipo de dato arreglos

```php
<?php

   $colores = array("rojo", "verde", "azul");
   $personas = array("nombre" => "Juan", "edad" => 30);

?>
```

Ejemplo de tipo de dato objetos

```php
<?php

   class Persona {
       public $nombre;
       public $edad;

       function __construct($nombre, $edad){
           $this->nombre = $nombre;
           $this->edad = $edad;
       }
   }

   $persona = new Persona("Juan", 30);

?>
```

Ejemplo de tipo de dato recursos

```php
<?php

   $archivo = fopen("datos.txt", "r");
   $conexion_db = mysqli_connect("localhost", "usuario", "clave",
"basededatos");

?>
```

Ejemplo de tipo de dato null

```php
<?php

   $valor_nulo = null;

?>
```

En resumen, PHP soporta varios tipos de datos, incluyendo booleanos, números enteros y de punto flotante, cadenas de caracteres, arreglos, objetos, recursos y null, lo que lo hace versátil y útil para una amplia gama de aplicaciones web.

Algo importante a saber es que PHP es un lenguaje de programación débilmente tipado (Loosely Typed Language). Esto significa que PHP asocia automáticamente un tipo de datos a la variable, dependiendo de su valor. Dado que los tipos de datos no se establecen en un sentido estricto, puede hacer cosas como agregar una cadena a un número entero sin causar un error.

En otras palabras, las variables en PHP pueden almacenar cualquier tipo de datos, sin tener que ser declaradas previamente con un tipo específico.

Por ejemplo, en un lenguaje de programación fuertemente tipado, como Java o C#, una variable debe ser declarada con un tipo específico antes de poder ser usada:

```
public class DatosPersonales {
   public static void main(String[] args) {
      String nombre = "Juan";
      int edad = 30;

      System.out.println("Mi nombre es " + nombre + " y tengo " + edad
+ " años.");
   }
}
```

En PHP, sin embargo, las variables no necesitan ser declaradas previamente con un tipo específico y pueden contener cualquier tipo de datos:

```php
<?php

   $numero = 42;
   $mensaje = "Hola mundo";

?>
```

Esta característica puede ser conveniente en algunos casos, pero también puede llevar a errores y confusiones si los programadores no prestan atención al tipo de datos que están manejando. Por lo tanto, aunque PHP es un lenguaje de programación débilmente tipado, es importante que los programadores comprendan y utilicen adecuadamente los tipos de datos en sus aplicaciones.

Strings

En programación, una "cadena" o "string" es una secuencia de caracteres, que pueden ser letras, números, símbolos o una combinación de ellos. Los strings se utilizan para representar información de texto, como nombres, palabras o frases. Las cadenas se delimitan entre comillas simples ('...') o comillas dobles ("...") en muchos lenguajes de programación, incluido PHP.

Funciones de strings

Las funciones de cadenas en PHP son un conjunto de funciones que se utilizan para manipular y trabajar con cadenas de texto. Estas funciones permiten realizar acciones como concatenar dos o más cadenas, buscar una subcadena dentro de una cadena, convertir una cadena a mayúsculas o minúsculas, recortar una parte de una cadena, calcular la longitud de una cadena, entre otras acciones.

Estas funciones son útiles en muchas situaciones, como cuando se trabaja con formularios de entrada de datos, se manipulan nombres de usuario, se procesan direcciones de correo electrónico, entre otros. Al utilizar estas funciones, es posible realizar operaciones complejas con cadenas de forma más sencilla y eficiente.

Algunos ejemplos de funciones de strings son:

- **strlen():** Devuelve la longitud de una cadena.
- **str_word_count():** Devuelve el número de palabras en una cadena.
- **strrev():** Invierte el orden de los caracteres en una cadena.
- **strpos():** Busca la primera aparición de una subcadena en una cadena y devuelve su posición.
- **str_replace():** Reemplaza una parte de una cadena con otra.
- **substr():** Devuelve una parte de una cadena.
- **strtolower():** Convierte una cadena a minúsculas.
- **strtoupper():** Convierte una cadena a mayúsculas.

- **trim():** Elimina los espacios en blanco al principio y al final de una cadena.
- **ltrim():** Elimina los espacios en blanco al principio de una cadena.
- **rtrim():** Elimina los espacios en blanco al final de una cadena.
- **explode():** Divide una cadena en un array en función de un delimitador.
- **implode():** Une los elementos de un array en una cadena, utilizando un delimitador especificado.
- **str_split():** Divide una cadena en un array de caracteres.
- **nl2br():** Inserta una etiqueta HTML
 antes de cada salto de línea en una cadena.
- **htmlentities():** Convierte todos los caracteres aplicables a entidades HTML.

*Ejemplo de uso de la función **strlen**:*

```php
<?php

   $str = "Hola mundo";
   echo strlen($str); // Salida: 11

?>
```

*Ejemplo de uso de la función **str_word_count**:*

```php
<?php

   $str = "Hola mundo";
   echo str_word_count($str); // Salida: 2

?>
```

*Ejemplo de uso de la función **strrev**:*

```php
<?php

   $str = "Hola mundo";
   echo strrev($str); // Salida: odnum aloH

?>
```

*Ejemplo de uso de la función **strpos**:*

```php
<?php

   $str = "Hola mundo";
   echo strpos($str, "mundo"); // Salida: 5

?>
```

*Ejemplo de uso de la función **str_replace**:*

```php
<?php

   $str = "Hola mundo";
   echo str_replace("mundo", "PHP", $str); // Salida: Hola PHP

?>
```

*Ejemplo de uso de la función **substr**:*

```php
<?php

   $str = "Hola mundo";
   echo substr($str, 5); // Salida: mundo

?>
```

*Ejemplo de uso de la función **strtolower**:*

```php
<?php

   $str = "HOLA MUNDO";
   echo strtolower($str); // Salida: hola mundo

?>
```

*Ejemplo de uso de la función **strtoupper**:*

```php
<?php

   $str = "hola mundo";
   echo strtoupper($str); // Salida: HOLA MUNDO

?>
```

*Ejemplo de uso de la función **trim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo trim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, se eliminaron los espacios de ambos lados.*

*Ejemplo de uso de la función **ltrim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo ltrim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, solo se eliminaron los espacios de la izquierda.*

*Ejemplo de uso de la función **rtrim:***

```php
<?php

   $str = "   Hola mundo   ";
   echo rtrim($str); // Salida: Hola mundo

?>
```

** En este ejemplo, solo se eliminaron los espacios de la derecha.*

*Ejemplo de uso de la función **explode:***

```php
<?php

   $str = "Hola mundo";
   $arr = explode(" ", $str);
   print_r($arr); // Salida: Array ( [0] => Hola [1] => mundo )

?>
```

*Ejemplo de uso de la función **implode:***

```php
<?php

   $arr = array("Hola", "mundo");
   $str = implode(" ", $arr);
   echo $str; // Salida: Hola mundo

?>
```

*Ejemplo de uso de la función **str_split:***

```php
<?php

   $str = "Hola mundo";
   $arr = str_split($str);
   print_r($arr);
   /* Salida:
    Array
    (
      [0] => H
      [1] => o
      [2] => l
      [3] => a
      [4] =>
      [5] => m
      [6] => u
      [7] => n
      [8] => d
      [9] => o
    )
  */

?>
```

*Ejemplo de uso de la función **nl2br:***

```php
<?php

   $str = "Hola mundo\nAdiós mundo";
   echo nl2br($str);

   /* Salida:
    Hola mundo
    Adiós mundo
   */

?>
```

Ejemplo de uso de la función ***htmlentities:***

```php
<?php

   $str = "<b>Hola mundo</b>";
   echo htmlentities($str); // Salida: &lt;b&gt;Hello World&lt;/b&gt;

?>
```

Estos son solo algunos de los métodos más comunes para el manejo de strings en PHP, hay muchos más disponibles y que se pueden encontrar en la sección PHP: Funciones de Strings en la documentación oficial del lenguaje.

Números

En programación, existen dos tipos principales de números: **enteros (integers) y flotantes (floats).**

1. **Números enteros:**
   un número entero es un número sin decimales, como -3, 0, 1, 100, etc. En la mayoría de los lenguajes de programación, los números enteros se representan mediante una secuencia de dígitos sin decimales. En PHP, se pueden representar tanto mediante decimales como hexadecimales.

2. **Números flotantes:**
   un número flotante es un número con decimales, como -3.14, 0.0, 1.23, 100.01, etc. Los números flotantes se representan mediante una secuencia de dígitos con un punto decimal, y se utilizan para representar números con una precisión más fina que los números enteros.

*Ejemplo de números enteros**:***

```php
<?php

   $int1 = 10;
   $int2 = -20;

   echo "Int1: $int1"; // Salida: Int1: 10
   echo "<br>Int2: $int2"; // Salida: Int2: -20

?>
```

*Ejemplo de números flotantes**:***

```php
<?php

   $float1 = 10.5;
   $float2 = -20.25;

   echo "Float1: $float1"; // Salida: Float1: 10.5
   echo "<br>Float2: $float2"; // Salida: Float2: -20.25

?>
```

Función is_numeric:
La función is_numeric en PHP se utiliza para determinar si un valor dado es un número o no. Devuelve TRUE si el valor es un número, y FALSE en caso contrario.

*Ejemplo de uso función **is_numeric:***

```php
<?php

   $num1 = "100";
   $num2 = 200;
   $num3 = "Hello";

   echo "Num1 is numeric: " . is_numeric($num1);
   // Salida: Num1 is numeric: 1

   echo "<br>Num2 is numeric: " . is_numeric($num2);
   // Salida: Num2 is numeric: 1

   echo "<br>Num3 is numeric: " . is_numeric($num3);
   // Salida: Num3 is numeric:

?>
```

Sin embargo, en el ejemplo anterior la variable $num1 contiene una cadena (“100”) pero de todas formas la función devuelve TRUE, esto se debe a que para que PHP pueda interpretar a esa variable como numérica es necesario antes convertirla a integer.

Entonces, aplicando la conversión de tipos de datos al ejemplo anterior, quedaría así:

```php
<?php

   $num1 = "100";

   if (is_numeric($num1)){
       $num1 = (int)$num1;
       echo "Num1 después de casteo: " . $num1 . "\n";
       // Salida: Num1 después de casteo: 100
   }else{
       echo "Num1 no es un número.\n";
   }
```

```
?>
```

Casteo

El casteo o casting en programación es la acción de convertir un valor de un tipo de datos a otro tipo de datos. En algunos lenguajes de programación, como PHP, el tipo de datos de una variable puede cambiar automáticamente en función de su contenido. Sin embargo, en otros casos, es necesario realizar una conversión explícita (o casteo) para cambiar el tipo de datos de una variable.

El casteo se puede realizar de forma implícita (cuando el lenguaje de programación lo hace automáticamente) o explícita (cuando se utiliza una función o un operador para realizar la conversión).

Algunos ejemplos de casteo en PHP son (int), (float), (string), (array), etc. Estos operadores se colocan antes de la variable que se quiere convertir y cambian su tipo de datos a la representación correspondiente (entero, flotante, cadena o array, respectivamente).

Ejemplo de casteo a entero:

```php
<?php

   $num1 = "100";
   $num2 = 200;
   $num3 = "Hello";

   $int1 = (int)$num1;
   $int2 = (int)$num2;
   $int3 = (int)$num3;

   echo "int1: " . $int1; // Salida: int1: 100
   echo "<br>int2: " . $int2; // Salida: int2: 200
   echo "<br>int3: " . $int3; // Salida: int3: 0

?>
```

Ejemplo de casteo a flotante:

```php
<?php

   $num1 = "100.5";
   $num2 = 200;
   $num3 = "Hello";

   $float1 = (float)$num1;
   $float2 = (float)$num2;
   $float3 = (float)$num3;

   echo "float1: " . $float1; // Salida: float1: 100.5
   echo "<br>float2: " . $float2; // Salida: float2: 200
   echo "<br>float3: " . $float3; // Salida: float3: 0

?>
```

Ejemplo de casteo a string:

```php
<?php

   $num1 = 100;
   $num2 = 200.5;
   $num3 = true;

   $string1 = (string)$num1;
   $string2 = (string)$num2;
   $string3 = (string)$num3;

   echo "string1: " . $string1; // Salida: string1: 100
   echo "<br>string2: " . $string2; // Salida: string2: 200.5
   echo "<br>string3: " . $string3; // Salida: string3: 1

?>
```

Vamos a explicar lo que sucedió.

En PHP, la palabra clave **true** se trata como un valor booleano, que se considera equivalente a 1 en comparaciones numéricas. Por lo tanto, al hacer un casteo explícito a string, el valor true se convierte en una cadena que representa el número 1. Esto se aplica a todos los valores booleanos en PHP.

En PHP, la palabra clave **false** se trata como un valor booleano que se considera equivalente a 0 en comparaciones numéricas. Sin embargo, al hacer un casteo explícito a string, el valor false se convierte en una cadena vacía (``), no en una cadena que representa el número 0. Esto se aplica a todos los valores booleanos en PHP.

Hemos llegado así al final de esta clase en la que vimos:

- Alcance o ámbito de las variables.
- Constructores del lenguaje echo y print.
- Tipos de datos.
- Strings.
- Funciones de strings.
- Números.
- Casteo.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP:
https://www.php.net/manual/es/index.php

W3Schools: PHP Tutorial:
https://www.w3schools.com/php/default.asp

Cabezas Granado, L., González Lozano, F., (2018). Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O., (2016). Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, L., Thompson, L., (2017). Desarrollo Web con PHP y MySQL. Quinta Edición. Editorial Anaya.

Para ampliar la información

PHP: Ámbito de las variables

PHP: Echo

PHP: Print

PHP: Tipos

PHP: Funciones de strings

PHP: Números enteros

PHP: Manipulación de tipos

Guía de HTML

Todo sobre PHP

PHP: The Right Way

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero