



PHP
+ OOP

PHP

Clase 13



- Type Hinting
- Return Type
- Clases
 1. Propiedades
 2. New
 3. ->
 4. Constructor
 5. Constantes
 6. Scope



PHP

Type Hinting



En la firma de una función, PHP nos permite aclarar qué tipo de datos se esperan.

Hasta PHP 7, sóloamente podíamos “hintear” arrays y clases. Desde PHP 7, también podemos “hintear” los tipos básicos.



Type Hinting

```
<?php
    function holaSoy(string $nombre, int $edad) {
        return "<h2>Hola soy ".$nombre." y tengo
        ".$edad." años";
    }
?>
```

De esta forma, nos aseguramos que ambos parámetros van a ser del tipo **que la función necesita**. Los tipos de datos que se pueden "Hintear" son: string, int, float, boolean, array, objeto*.

PHP

Return Types



En PHP 7, una nueva característica ha sido implementada: **Declarar el Tipo de Retorno**. La declaración *Return type* especifica el tipo de dato que una función debe retornar.





Return Types

- int
- float
- bool
- string
- Interfaces y clases
- array
- callable

Agregando un “?” podemos indicar que se podría devolver null.



Return Types

```
<?php
    declare(strict_types = 1);

    function returnIntValue(int $value): int {
        return $value;
    }
    print(returnIntValue(5));
?>
```

Devuelve 5.



Return Types

```
<?php
    declare(strict_types = 1);

    function returnIntValue(int $value): int {
        return $value + 1.0;
    }
    print(returnIntValue(5));
?>
```

Fatal error: Uncaught TypeError: Return value of returnIntValue() must be of the type integer, float returned...



Return Types

```
<?php
declare(strict_types = 1);

function returnValue(int $value): ?int {
    return $value > 0 ? $value : null;
}

echo returnValue(5); // imprime 5
echo returnValue(-5); // imprime vacío
```

PHP

Clases



Una clase es un molde para la creación de objetos. Definen un conjunto de propiedades, estados y el comportamiento de dicha entidad, mediante sus métodos.



Class

Se declara como:

```
<?php  
    class Usuario {  
  
    }  
?>
```




Class - Propiedades

```
<?php
    class Usuario {
        public $nombre;
        public $email;
        public $contraseña;
    }
?>
```



->

El operador -> nos permite acceder a un atributo o un método de la instancia de un objeto.





new

```
<?php
    $usu1 = new Usuario();
    $usu1->nombre = "Agus";
    $usu1->email = "Agus@digitalhouse.com";

    $usu2 = new Usuario();
    $usu2->email = "admin@admin.com";
    $usu2->contraseña = "admin123";
?>
```



Constructor

Los constructores son funciones en una clase que son invocadas automáticamente cuando se crea una nueva instancia de una clase con **new**.

En php se define como:

```
public function __construct( )
```

También puede ser privado...

Constructor

```
<?php
    class Usuario {
        public $nombre;
        public function __construct($nombrePersona) {
            $this->nombre = $nombrePersona;
        }
    }
    $usu1 = new Usuario("Pepe");
    echo $usu1->nombre; //esto imprime Pepe
    $usu2 = new Usuario("Juan");
    var_dump( $usu2 ) //esto imprime Juan
?>
```


¡A practicar!



```
<?php  
    echo "Hora de practicar!";  
?>
```

Class - Métodos

```
<?php
    class Usuario {
        private $contraseña;
        public function getPass() {
            return $this->contraseña;
        }
        public function setPass($pass) {
            $this->contraseña = $pass;
        }
    }
?>
```



Class - Métodos

El uso de **`this`** dentro de un método referencia a **la instancia puntual en donde será ejecutada el método.**




Class - Métodos

```
<?php
    include("usuario.php");

    $usu1 = new Usuario();

    $usu1->setPass("12345");

    echo $usu1->getPass(); // imprime 12345
?>
```





Constantes de clase

```
<?php
    class Prefijo {
        const BUENOS_AIRES = '011';
        const MAR_DEL_PLATA = '0223';
    }

    echo Prefijo::BUENOS_AIRES;
?>
```

PHP

Scope



Dentro de una clase, los distintos atributos y métodos tienen diferentes alcances:

- public
- private
- protected





Scope public

```
<?php
    class Usuario {
        public $nombre;
    }


    $usu1 = new Usuario();
    $usu1->nombre = "Juan";
?>
```

Esto vale!



Scope public

El modificador public hace esa propiedad visible desde cualquier entorno en PHP.





Scope private

```
<?php
    class Usuario {
        private $nombre;
    }

    $usu1 = new Usuario();
    $usu1->nombre = "Juan";
?>
```


Esto no vale!





Scope private

El modificador private hace que esa propiedad sea únicamente visible desde la clase a la que pertenece.



Scope protected

```
<?php
    class Usuario {
        protected $nombre;
    }

    $usu1 = new Usuario();
    $usu1->nombre = "Juan";
?>
```

Esto no vale!



Scope protected

El modificador protected hace que esa propiedad sea únicamente visible desde la clase a la que pertenece **y de sus clases hijas**.

Veremos más adelante cómo implementar protected de manera funcional cuando veamos el concepto de herencia en PHP.



Class - Métodos

¡Los métodos pueden tener los mismos scopes que los atributos!



Class - Constantes

¡Las constantes también pueden ser públicas (default), privadas o protegidas!



¡A practicar!



```
<?php  
    echo "Hora de practicar!";  
?>
```

¡Gracias!



¿Preguntas?