





# Type Hinting

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

Hasta PHP 7, sólo podíamos “hintear” arrays y Clases.

Desde PHP 7, también podemos “hintear” los tipos básicos.





# Type Hinting

```
<?php
```

```
function nombreCompleto(string $nombre, string $Apellido) {  
    return $nombre . " de " . $apellido;  
}
```

```
?>
```

De esta forma, aseguro que ambos parámetros van a ser de tipo **string**.






# Type Hinting

```
<?php
```

```
function apellidoCasada(Persona $ella, Persona $el) {  
    return $ella->getApellido() . " de " . $el->getApellido();  
}
```

```
?>
```

De esta forma, aseguro que ambos parámetros van a ser de tipo **Persona** y puedo asegurar que cuentan con dichos métodos.





# 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. Los siguientes tipos de datos pueden ser declarados:





# Return Types

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





# 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...





# 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.



A decorative pattern of hexagons in various shades of blue and cyan on the left side of the slide. Some hexagons contain icons: a lightbulb, a thumbs up, a network of nodes, a smartphone, a magnifying glass, a gear, and a speech bubble.

# Class

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



# Class – Propiedades

```
<?php
    class Auto
    {
        public $color;
        public $marca;
        public $modelo;
    }
?>
```



# new

```
<?php
    $auto1 = new Auto();
    $auto1->marca = "Chevrolet";
    $auto1->modelo = "Corsa";
```

```

    $auto2 = new Auto();
    $auto2->marca = "Renault";
    $auto2->modelo = "Sandero";
```

```
?>
```



→

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



new

```
<?php
    echo "Tengo un {$auto1->marca} {$auto1->modelo}";

    echo "Tengo un {$auto2->marca} {$auto2->modelo}";
?>
```



# 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

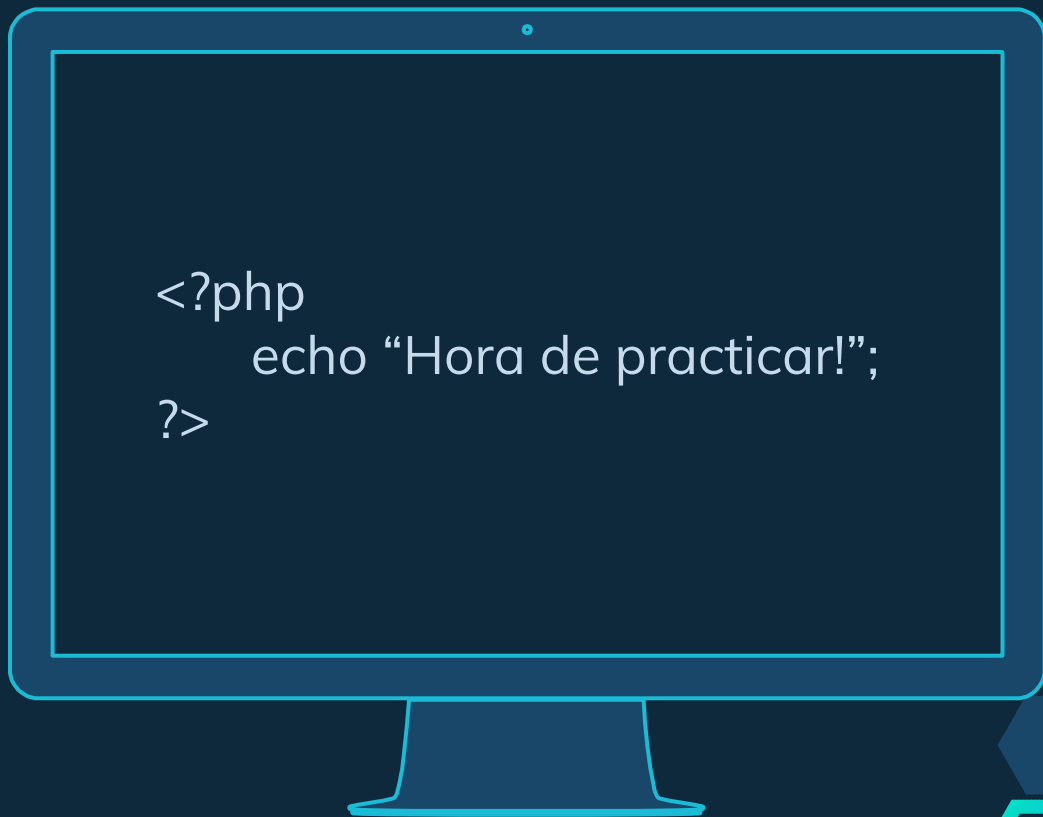
```
<?php
    class Persona {
        public $nombre;
        public function __construct($nombrePersona) {
            $this->nombre = $nombrePersona;
        }
    }
    $persona1 = new Persona("Pepe");
    echo $persona1->nombre; //esto imprime Pepe
    $persona2 = new Persona("Juan");
    echo $persona2->nombre; //esto imprime Juan
?>
```





# ¡A practicar!

Ejercicios del 1 al 8





# Constantes de clase

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

    echo Prefijo::BUENOS_AIRES;

?>
```



define



const

Puede definirse en condicionales	No puede definirse en condicionales
Por default es case sensitive	Case sensitive
Es una función de PHP, consume recursos	Construcción del lenguaje
No se puede utilizar para clases	Se puede utilizar para clases



# Scope

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

- ◇ public
- ◇ private
- ◇ protected





# Scope public

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

    $persona = new Persona();
    $persona->nombre = "Juan";

?>
```

Esto vale!

A decorative graphic on the left side of the slide. It features a large, solid blue hexagon in the center. Surrounding it are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There are also some abstract shapes like a network of dots and a speech bubble.

# Scope public

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



# Scope private

```
<?php
    class Persona
    {
        private $nombre;

        $persona = new Persona();
        $persona->nombre = "Juan";
    }
?>
```

Esto no vale!

A decorative graphic on the left side of the slide. It features a large, central cyan hexagon. Surrounding it are several smaller hexagons in various shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with three nodes and a speech bubble icon.

# Scope private

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





# Scope protected

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

    $persona = new Persona();
    $persona->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

```
<?php
class Auto
{
    private $color;
    public function getColor() {
        return $this->color;
    }
    public function setColor($color) {
        $this->color = $color;
    }
}
?>
```

A decorative graphic on the left side of the slide. It features a large cyan hexagon in the center. Surrounding it are several smaller hexagons of different shades of blue and cyan. Some of these hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and connecting lines.

# 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("auto.php");
```

```
$auto = new Auto();
```

```
$auto->setColor("Negro");
```

```
echo $auto->getColor();
```

```
?>
```

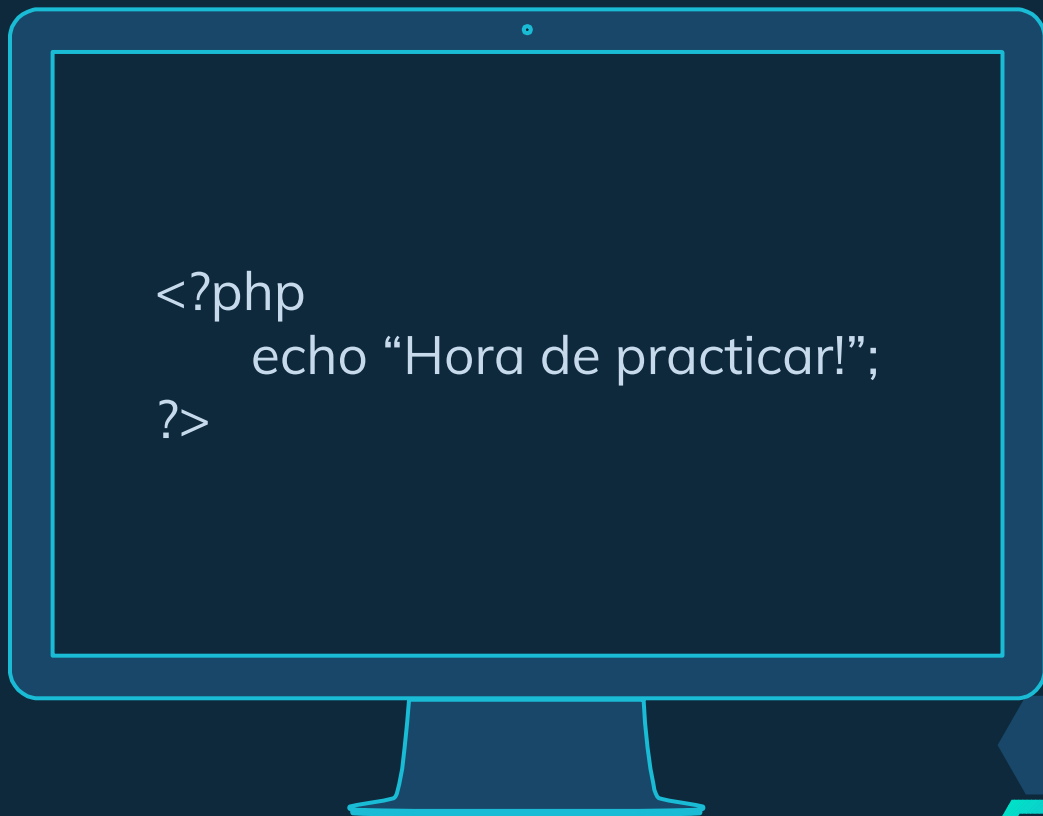
A decorative graphic on the left side of the slide. It features a large, solid blue hexagon in the center. Surrounding it are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There are also some abstract shapes like a network of dots and a speech bubble.

# Class – Métodos

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



¡A practicar!





# Gracias!

¿Preguntas?

