PHP - Type Hints

PHP supports using "type hints" at the time of declaring variables in the function definition and properties or instance variables in a class. PHP is widely regarded as a weakly typed language. In PHP, you need not declare the type of a variable before assigning it any value.

The PHP parser tries to cast the variables into compatible type as far as possible. Hence, if one of values passed is a string representation of a number, and the second is a numeric variable, PHP casts the string variable to numeric in order to perform the addition operation.

Example

Take a look at the following example −

Open Compiler

```
<?php
   function addition($x, $y) {
      echo "First number: $x Second number: $y Addition: " . $x+$y;
   }
   $x="10";
   $y=20;
   addition($x, $y);
?>
```

It will produce the following **output** −

```
First number: 10 Second number: 20 Addition: 30
```

However, if **$x** in the above example is a string that doesn't hold a valid numeric representation, then you would encounter an error.

Open Compiler

```
<?php
   function addition($x, $y) {
      echo "First number: $x Second number: $y Addition: " . $x+$y;
   }
   $x="Hello";
   $y=20;
   addition($x, $y);
?>
```

It will produce the following **output** –

```
PHP Fatal error:  Uncaught TypeError: Unsupported operand types: string + int in hello.p
```

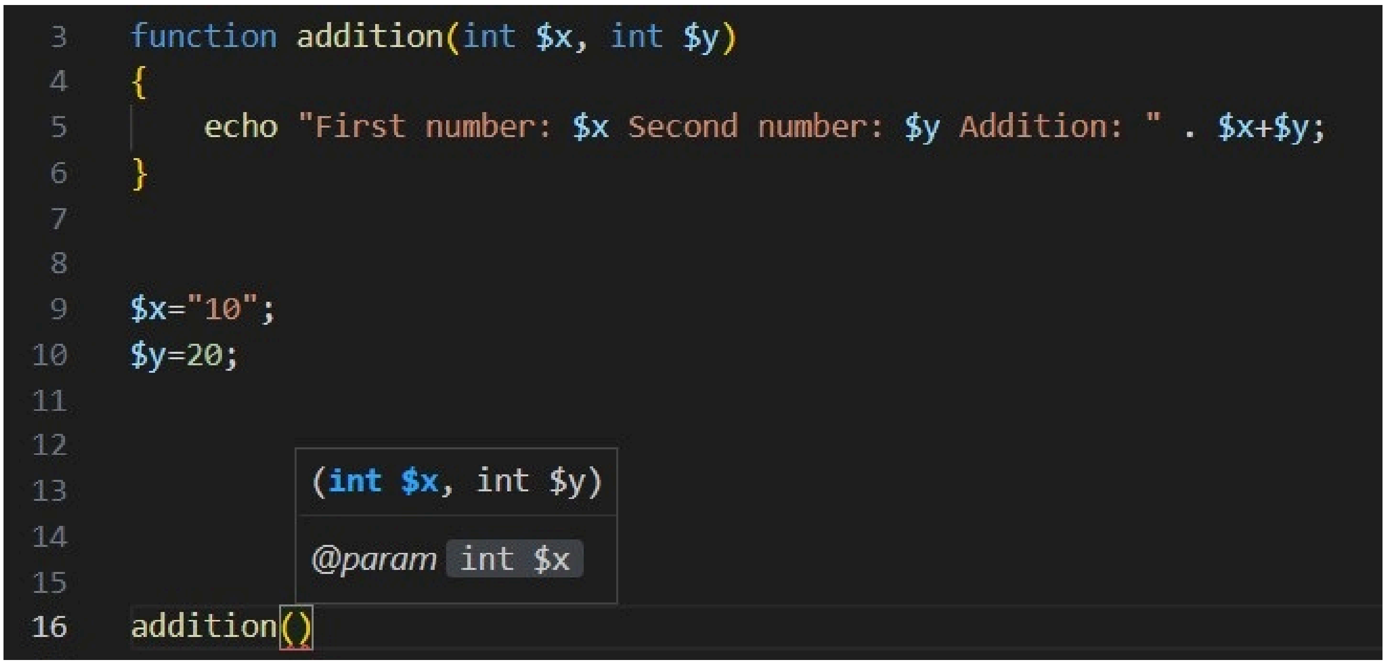
Type-hinting is supported from PHP 5.6 version onwards. It means you can explicitly state the expected type of a variable declared in your code. PHP allows you to type-hint function arguments, return values, and class properties. With this, it is possible to write more robust code.

Let us incorporate type-hinting in the addition function in the above program –

```
function addition($x, $y) {
   echo "First number: $x Second number: $y Addition: " . $x+$y;
}
```

The type-hinting feature is mostly used by IDEs (Integrated Development Environment) to prompt the user about the expected types of the parameters used in function declaration.

The following figure shows the VS Code editor popping up the function prototype as you type –



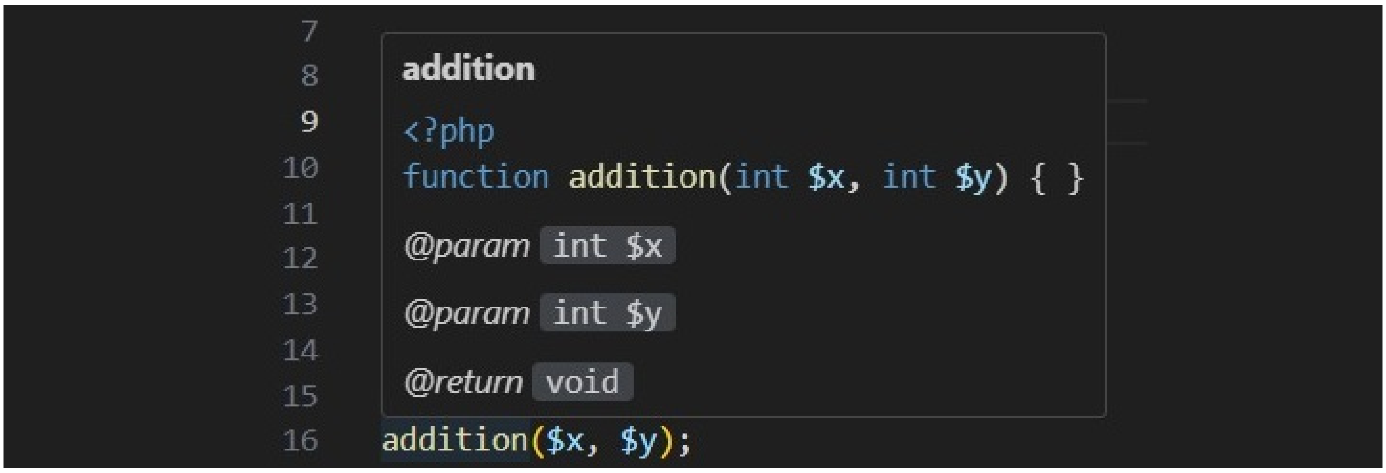
```
function addition(int $x, int $y)
{
    echo "First number: $x Second number: $y Addition: " . $x+$y;
}


$x="10";
$y=20;




addition()
```

If the cursor hovers on the name of the function, the type declarations for the parameters and the return value are displayed −



```
addition
<?php
function addition(int $x, int $y) { }
@param int $x
@param int $y
@return void
addition($x, $y);
```

Note that by merely using the data types in the variable declarations doesn’t prevent the unmatched type exception raised, as PHP is a dynamically typed language. In other words, $x="10" and $y=20 will still result in the addition as 30, where as $x="Hello" makes the parser raise the error.

strict_types

PHP can be made to impose stricter rules for type conversion, so that "10" is not implicitly converted to 10. This can be enforced by setting strict_types directive to 1 in a declare() statement. The declare() statement must be the first statement in the PHP code, just after the "**<?php**" tag.

Example

Open Compiler

```
<?php
   declare (strict_types=1);
   function addition(int $x, int $y) {
      echo "First number: $x Second number: $y Addition: " . $x+$y;
   }
   $x=10;
   $y=20;
   addition($x, $y);
?>
```

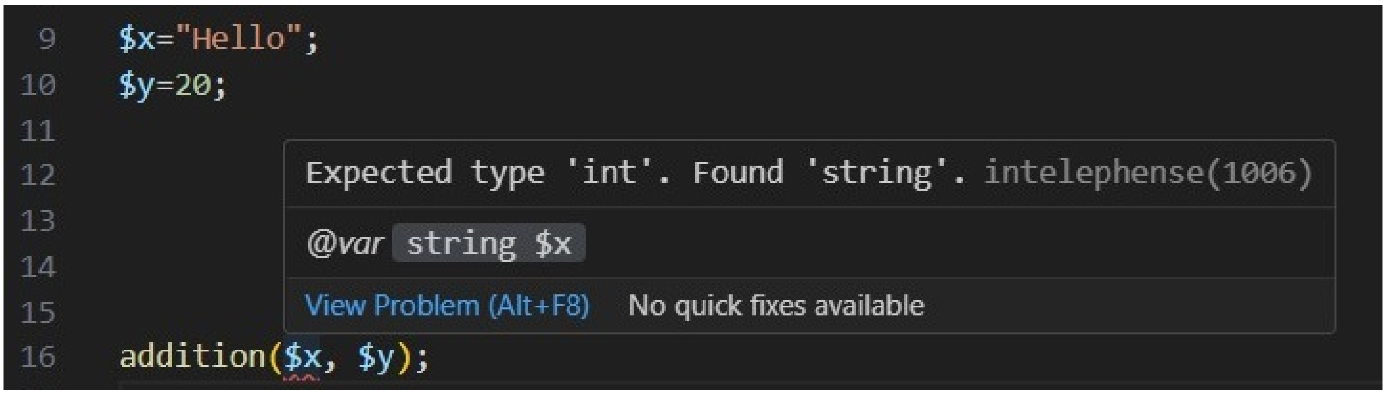
It will produce the following **output** −

```
First number: 10 Second number: 20 Addition: 30
```

Now, if **$x** is set to "10", the implicit conversion wont take place, resulting in the following error −

```
PHP Fatal error:  Uncaught TypeError: addition(): Argument #1 ($x) must be of type int,
```

The VS Code IDE too indicates the error of the same effect −



```
$x="Hello";
$y=20;

                Expected type 'int'. Found 'string'. intelephense(1006)
                @var string $x
                View Problem (Alt+F8)   No quick fixes available

addition($x, $y);
```

From PHP 7 onwards with type-hinting support has been extended for function returns to prevent unexpected return values. You can type-hint return values by adding the intended type after the parameter list prefixed with a colon (:) symbol.

Example

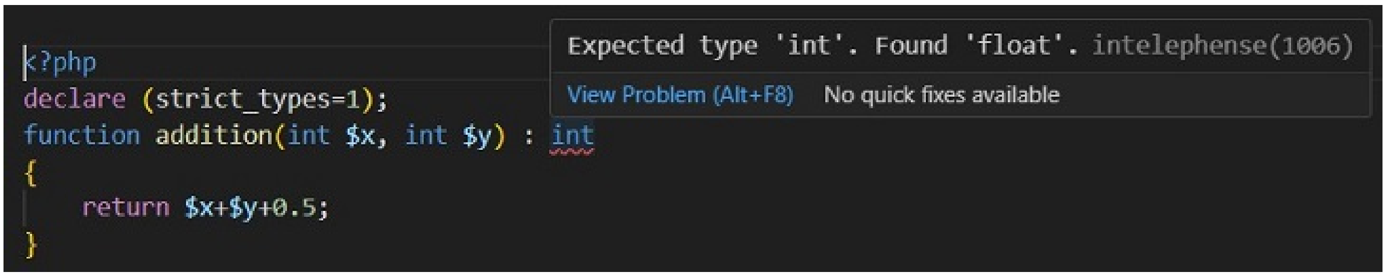
Let us add a type hint to the return value of the addition function above −

```php
<?php
   declare (strict_types=1);
   function addition(int $x, int $y) : int {
      return $x+$y;
   }

   $x=10;
   $y=20;

   $result = addition($x, $y);
   echo "First number: $x Second number: $y Addition: " . $result;
?>
```

Here too, if the function is found to return anything other than an integer, the IDE indicates the reason even before you run.



```
<?php
declare (strict_types=1);
function addition(int $x, int $y) : int
{
    return $x+$y+0.5;
}
```

```
Expected type 'int'. Found 'float'. intelephense(1006)
View Problem (Alt+F8)    No quick fixes available
```

Union Types

The PHP introduced union types with its version 8.0. You can now specify more than one type for a single declaration. The data types are separated by the "**|**" symbol.

Example

In the definition of addition() function below, the **$x** and **$y** arguments can be of **int** or **float** type.

Open Compiler

```
<?php
   declare (strict_types=1);
   function addition(int|float $x, int|float $y) : float {
      return $x+$y;
   }
   $x=10.55;
   $y=20;

   $result = addition($x, $y);
   echo "First number: $x Second number: $y Addition: " . $result;
?>
```

Type-hinting in Class

In PHP, from version 7.4 onwards, you can use the type hints in declaration of class properties and methods.

Example

In the following example, the class constructor uses type hints −

</> Open Compiler

```php
<?php
   declare (strict_types=1);
   class Student {
      public $name;
      public $age;
      public function __construct(string $name, int $age) {
         $this->name = $name;
         $this->age = $age;
      }

      public function dispStudent() {
         echo "Name: $this->name Age: $this->age";
      }
   }
   $s1 = new Student("Amar", 21);
   $s1->dispStudent();
?>
```

It is also possible to use type hints in the declaration of class properties.

```php
class Student {
   public string $name;
   public int $age;

   public function __construct($name, $age) {
      $this->name = $name;
      $this->age = $age;
   }

   public function dispStudent() {
      echo "Name: $this->name Age: $this->age";
   }
}
```

The most commonly encountered errors during program development are **type errors**. The type-hinting feature helps in reducing them.