PHP is_null() Function

PHP defines NULL as one of its special data types. It indicates that a certain variable has not been assigned a value any specific data type. It is a built-in constant in PHP and is used to indicate the intentional absence of any object or value. A variable can be explicitly assigned NULL or its value been set to null by using the **unset()** function.

The is_null() Function

PHP provides a Boolean function **is_null()** to check if a variable is indeed of NULL type.

```
is_null(mixed $value): bool
```

Example 1

If any variable is explicitly assigned NULL, obviously the **is_null()** function returns **true**.

Open Compiler

```
<?php
   $x = NULL;
   echo "Variable \$x is null? ";
   var_dump(is_null($x));
?>
```

It will produce the following **output** −

```
Variable $x is null? bool(true)
```

Example 2

If a variable with a certain value is unset, then too the is_null() function returns true, but with a warning

Open Compiler

```
<?php
   $x = "Hello";
   unset($x);
   echo "Variable \$x is null?\n";
   var_dump(is_null($x));
?>
```

It will produce the following **output** −

```
Variable $x is null?
bool(true)

PHP Warning:  Undefined variable $x in /home/cg/root/89262/main.php on line 5
```

Example 3

Similarly, if you just declare a variable, without assigning any value to it, the is_null() function returns **true** with a warning −

Open Compiler

```
<?php
   $y;
   echo "Variable \$y is null?\n";
   var_dump(is_null($y));
?>
```

It will produce the following **output** −

```
Variable $y is null?
bool(true)
Warning: Undefined variable $y in hello.php on line 9
```

Example 4

You can also use the equality operator (==) to check if a variable is NULL.

Open Compiler

```
<?php
   $x = NULL;
   if ($x === NULL) {
      echo '$x is NULL';
   } else {
      echo '$x is not NULL';
   }
?>
```

It will produce the following **output** −

```
$x is NULL
```

Example 5

A null string "" is not considered equal to NULL. Hence, the is_null() function as well as the "==" operator return **false**. Take a look at the following example −

</> Open Compiler

```
<?php
   $y = "";
   if ($y === NULL) {
      echo '$y is NULL';
   } else {
      echo '$y is not NULL';
   }
   echo "$y is null?\n";
   var_dump(is_null($y));
?>
```

It will produce the following **output** −

```
$y is not NULL is null?
bool(false)
```

Two other functions in PHP that are relevant to is_null() function are the **isset()** function and the **empty()** function.

The isset() Function

The isset() function determines if a variable is declared and is different than NULL.

```
isset(mixed $var, mixed ...$vars): bool
```

Example

A variable that is assigned NULL is considered as unset.

</> Open Compiler

```
<?php
   $x = NULL;
   echo '$x is set? ';
   var_dump(isset($x));
?>
```

It will produce the following **output** −

```
$x is set? bool(false)
```

Note that a null character ("\0") is not equivalent to the PHP null constant.

The empty() Function

The empty() function checks if a variable is considered to be empty. A variable is considered empty if it does not exist or if its value is NULL. empty() does not generate a warning if the variable does not exist.

Example 1

Take a look at the following example −

</> Open Compiler

```
<?php
   $x = NULL;
   echo '$x is empty? ';
   var_dump(empty($x));
   $y;
   echo '$y is empty? ';
   var_dump(empty($y));
?>
```

It will produce the following **output** –

```
$x is empty? bool(true)
$y is empty? bool(true)
```

Example 2

The empty() function returns **true** if a variable is set to "0", NULL, or is not set at all.

</> Open Compiler

```
<?php
   $var = 0;
   if (empty($var)) {
      echo '$var is either 0, empty, or not set at all';
   }
?>
```

It will produce the following **output** –

```
$var is either 0, empty, or not set at all
```