Programming Basics

Vivify Academy

VIVIFY
ACADEMY

Variable Scope

Understanding Variable Scope

```php
<?php

function test()
{
    $greet = "Hello World!";
    echo $greet;
}

test(); // Outputs: Hello World!

// Generate undefined variable error
echo $greet;

?>
```

You can declare variables anywhere in a PHP script. However, the location of the declaration determines where the variable can be used within the PHP program, in other words its availability. This accessibility is known as the *scope* of a variable.

By default, variables declared within a function are *local* and they cannot be viewed or manipulated from outside of that function, as demonstrated in the example. They exist from the moment when the function is called, and disappear when the function ends.

Understanding Variable Scope

```php
<?php

$greet = "Hello World!";

function test()
{
    echo $greet;
}


test(); // Generate undefined variable error

echo $greet; // Outputs: Hello World!

?>
```

Similarly, if you try to access an outside variable inside the function, you'll get an undefined variable error, as shown in the example.

Most programming languages follow this rule. Limiting the scope of a variable inside the function allows us to more easily understand what’s happening inside a function, because we don’t have to worry about what values the variables had before we called the function, or whether we changed some variable outside.

Understanding Variable Scope

```php
<?php

$greet = "Hello World!";

function test($greet)
{
    echo $greet;
}


test("Hello Mars!"); // Outputs: Hello Mars!

echo $greet; // Outputs: Hello World!

?>
```

Here we have example with script variable called $greet and the function variable called exactly the same.

Those two are different variables.

The first has a scope within the script (the whole program).

The second one has a scope only inside the function.

Global variables

```php
<?php
$greet = "Hello World!";

function test()
{
    global $greet;
    echo $greet;
}

test(); // Outputs: Hello World!
echo $greet; // Outputs: Hello World!

// Assign a new value to variable
$greet = "Goodbye";

test(); // Outputs: Goodbye
echo $greet; // Outputs: Goodbye
?>
```

There may be a situation when you need to import a variable from the main program into a function, or vice versa.

In such cases, you can use the **global** keyword, then the variable name.

This keyword turns the variable into a global variable, making it visible or accessible both inside and outside the function, as shown in the example.

What will happen here?

```
function triple ($x)
{
        $x = $x + $x + $x;
        return $x;
}

$x = 5;
echo triple ($x);
```

Constants

Constants in PHP

A constant is a name for a value that’s defined only once. By default, a constant name is case-sensitive. Constant names are most often written in uppercase letters, with underscores (_) separating words.

A constant value cannot change during the execution of the script. If you have defined a constant, it can never be changed or undefined.

Creating a constant

```php
<?php

define("MIN_SIZE", 50);
const MAX_SIZE = 1000;

echo constant("MIN_SIZE");
echo MIN_SIZE;

?>
```

To define a constant you can use the **const** keyword, or call the **define()** function, which takes the name of the constant (as a string) and the value of the constant.

To read a constant's value, you can simply use the constant’s name in place of any value, or call the **constant()** function with the name of the constant as a string.

Differences between constants and variables

- A dollar sign ($) before a constant will cause an error, whereas for variables you have to write a dollar sign.
- Constants don’t follow the scoping rules like those for variables; you can use a constant inside a function even if it was defined outside.
- You can also define a constant inside a function and use it outside, but only with the **define()** function, not with **const**.
- Once the constants have been set, they may not be redefined or undefined.

Declaring an array as constant

```php
<?php

// in php 5.6, you can declare an array constant with const
const RANDOM_NUMBERS = [3, 9, 1, 42];


// in php 7, you can finally use define()
define('RANDOM_NUMBERS', [3, 9, 1, 42]);

?>
```

Useful links

- http://php.net/manual/en/language.variables.scope.php
- http://php.net/manual/en/language.constants.php
- http://php.net/manual/en/language.constants.syntax.php

Exercise!

VIVIFY ACADEMY