FUNCTIONS – ARRAYS

In PHP, a function is a block of reusable code that performs a specific task. Functions help organize code, make it more modular, and facilitate code reuse.

WHAT IS A FUNCTION ?

You can define a function using the `function` keyword, followed by the function name, parameters (if any), and the code block.

```
function functionName($parameter1, $parameter2) {
   // Code to be executed
   return $result; // Optional: Return a value
}
```

Example:

```
function greet($name) {
   echo "Hello, $name!";
}
// Call the function
greet("John");
```

Function Parameters:

Parameters are variables passed to the function. You can define a function with or without parameters.

```
function add($num1, $num2) {
   return $num1 + $num2;
}

$result = add(5, 3);
```

Default Parameter Values:

You can assign default values to parameters, making them optional.

```
function greet($name = "Guest") {
   echo "Hello, $name!";
}
greet();       // Outputs: Hello, Guest!
greet("John");  // Outputs: Hello, John!
```

Returning Values:

Functions can return values using the `return` statement.

```
function square($number) {
   return $number * $number;
}
$result = square(4); // $result is 16
```

Variable Scope:

Variables defined inside a function have local scope by default. They are only accessible within that function.

```
function example() {
    $localVariable = "I am local!";
    echo $localVariable;
}
example(); // Outputs: I am local!
// echo $localVariable; // Error: $localVariable is not defined here
```

Global Variables:

To access a global variable within a function, use the `global` keyword.

```
$globalVariable = "I am global!";
function printGlobal() {
    global $globalVariable;
    echo $globalVariable;
}
printGlobal(); // Outputs: I am global!
```

Anonymous Functions (Closures):

Anonymous functions, also known as closures, allow you to create functions on-the-fly.

```
$add = function ($a, $b) {
    return $a + $b;
};
$result = $add(3, 5); // $result is 8
```

Variable Functions:

PHP allows you to use a variable as a function name.

```
function sayHello() {
    echo "Hello!";
}
$functionName = "sayHello";
$functionName(); // Outputs: Hello!
```

These are fundamental concepts related to functions in PHP. Functions play a crucial role in structuring code and promoting code reusability. They make it easier to maintain and organize your PHP applications.

ARRAYS

In PHP, an array is a versatile and essential data structure that allows you to store and manipulate multiple values under a single variable name. Arrays can hold various types of data, including numbers, strings, and even other arrays.

Indexed Arrays: Indexed arrays use numeric indices to access elements. The index starts from 0 and goes up to the number of elements minus one.

Creating Indexed Arrays:

```
$colors = array("red", "green", "blue");
```

or, using short syntax (PHP 5.4 and later):

```
$colors = ["red", "green", "blue"];
```

Accessing Elements:

```
echo $colors[0]; // Outputs: red
echo $colors[1]; // Outputs: green
echo $colors[2]; // Outputs: blue
```

Modifying Elements:

```
$colors[1] = "yellow";
```

Adding Elements:

```
$colors[] = "orange";
```

Associative Arrays:

Associative arrays use named keys to access elements instead of numerical indices. Each element has a key-value pair.

Creating Associative Arrays:

```
$user = array(
   "name" => "John",
   "age" => 25,
   "city" => "New York"
);
```

or, using short syntax:

```
$user = [
   "name" => "John",
   "age" => 25,
   "city" => "New York"
];
```

****Accessing Elements:****

```
echo $user["name"]; // Outputs: John
echo $user["age"];  // Outputs: 25
echo $user["city"]; // Outputs: New York
```

****Modifying Elements:****

```
$user["age"] = 26;
```

****Adding Elements:****

```
$user["gender"] = "Male";
```

Multi-dimensional Arrays:

A multi-dimensional array is an array that contains one or more arrays. These arrays can be indexed or associative.

****Creating Multi-dimensional Arrays:****

```
$matrix = array(
   array(1, 2, 3),
   array(4, 5, 6),
   array(7, 8, 9)
);
```

****Accessing Elements:****

```
echo $matrix[1][2]; // Outputs: 6
```

Array Functions:

PHP provides a variety of built-in functions for working with arrays, such as `count`, `array_push`, `array_pop`, `array_merge`, `array_slice`, and many more.

****Example: Using `count` function:****

```
$numbers = [1, 2, 3, 4, 5];
$count = count($numbers); // $count is 5
```

****Example: Using `array_push` and `array_pop`:****

```
$stack = [1, 2, 3];
array_push($stack, 4, 5); // $stack becomes [1, 2, 3, 4, 5]
$lastElement = array_pop($stack); // $lastElement is 5, $stack becomes [1, 2, 3, 4]
```

These are the basics of working with arrays in PHP. Arrays are fundamental to handling collections of data and are extensively used in PHP programming.