# PHP
# Array & Functions

# Array

- Array is a special variable, which can store multiple values under a single variable name.

- In PHP, there are three types of arrays:
  - **Indexed arrays** - Arrays with a numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays

# Indexed Arrays

- An indexed or numeric array stores each array element with a numeric index.
- The following examples shows two ways of creating an indexed array

- Example

```
<?php
$colors = array("Red", "Green", "Blue");
<?
```

Or using:

```
<?php
$colors[0] = "Red";
$colors[1] = "Green";
$colors[2] = "Blue";
<?
```

# Example:

```
<html>
    <head>
        <meta charset="UTF-8">
        <title> Array Example</title>
    </head>
    <body>
        <?php
        $colors = array("Red", "Green", "Blue");
        echo "my favourite clolor is " . $colors[2];
        ?>

    </body>
</html>
```

# The count() Function

- The count() function is used to return the number of elements of an array

```
<!DOCTYPE html>
<!--
count.php
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>Count Example</title>
    </head>
    <body>
        <?php
        $colors = array("Red", "Green", "Blue");
        echo "Number of elements in colors array is: ", count($colors);
        ?>
    </body>
</html>
```

# Example:

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $colors = array("Red", "Green", "Blue");
        $arrlength = count($colors);
        for($i=0; $i< $arrlength ; $i++){
        echo $colors[$i];
        echo"<br/>";
        }
        ?>
    </body>
</html>
```

# Associative Arrays

- Associative array are arrays that use named keys, these keys are user defined strings.
- In the following example the array uses keys instead of index numbers:

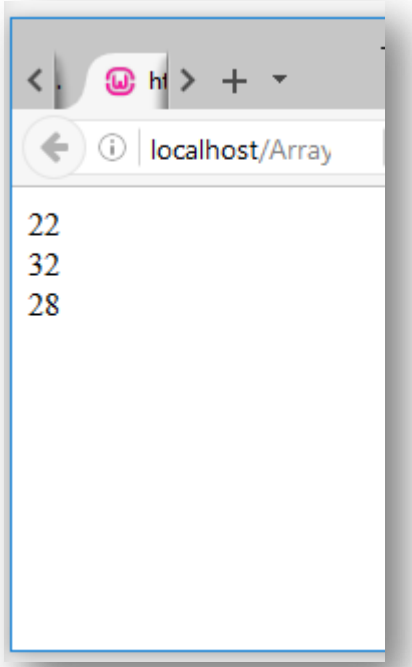
```
<?php
    $age = array("Ali" => "22", "Salma" => "32", "Khalid" => "28");
<?
```

Or using:

```
<?php
    $ages["Ali"] = "22";
    $ages["Salma"] = "32";
    $ages["Khalid"] = "28";
<?
```

# Example:

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $age = array("Ali" => "22", "Salma" => "32", "Khalid" => "28");
        foreach ($age as $name) {
            echo $name ;
            echo "<br>";
        }
        ?>
    </body>
</html>
```

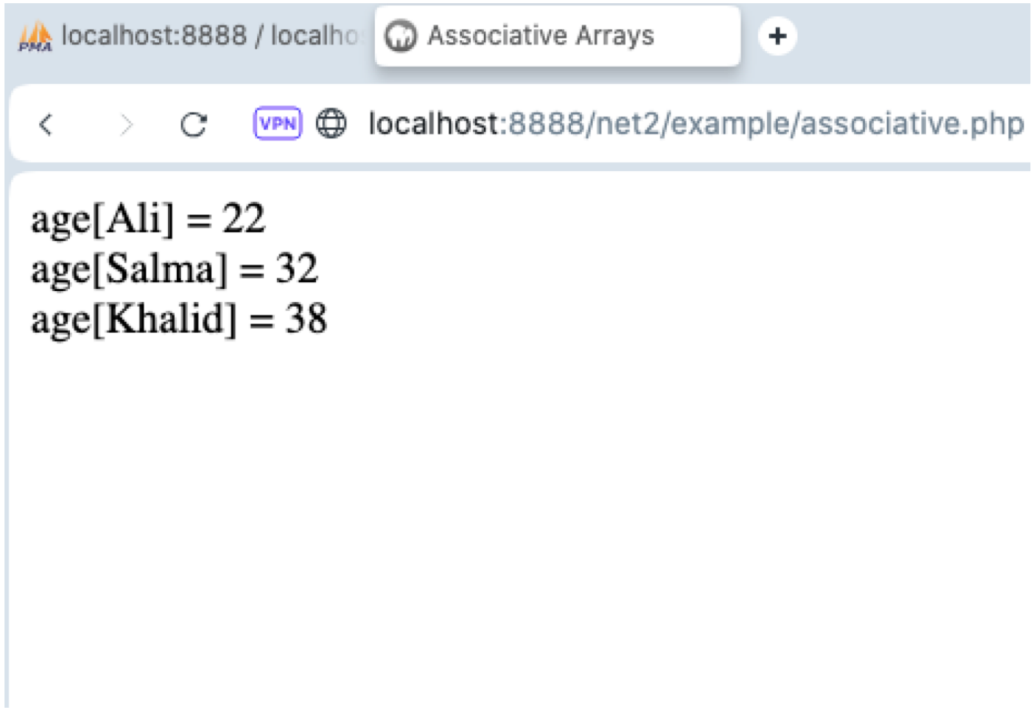


# Example:

associative.php

```php
<html>
  <head>
    <title>Associative Arrays</title>
  </head>
  <body>
    <?php
      $age = array("Ali" => 22, "Salma" => 32, "Khalid" => 38);
      foreach ($age as $name => $year){
          echo "age[".$name."] = ".$year."<br>";
      }
    ?>
  </body>
</html>
```

# associative.php result



# Multidimensional Arrays

- The multidimensional array is an array in which each element can also be an array and each element in the sub-array can be an array.

```
$contacts = array(
        array("Ahmad","Ahmad@mail.com" ),
        array("Salma", "Salma@mail.com"),
        array("Ali","Ali@mail.com")
);
```

# Example:

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $contacts = array(
            array("Ahmad","Ahmad@mail.com" ),
            array("Salma", "Salma@mail.com"),
            array("Ali","Ali@mail.com")
         );
        for ($row = 0; $row < 3; $row++) {
            for ($col = 0; $col < 2; $col++) {
                echo $contacts[$row][$col] ;
            }
           echo "<br/>";
        }
        ?>
    </body>
</html>
```

# Sorting Arrays

- PHP uses a number of built-in functions designed specifically for sorting array elements in different ways like alphabetically or numerically in ascending or descending order.

  - sort() – sort indexed  arrays in ascending order
  - rsort() - sort indexed arrays in descending order
  - asort() - sort associative arrays in ascending order, according to the value
  - ksort() - sort associative arrays in ascending order, according to the key
  - arsort() - sort associative arrays in descending order, according to the value
  - krsort() - sort associative arrays in descending order, according to the key

# Example:

- Sorting Indexed Arrays in Ascending Order

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $colors = array("Red", "Green", "Blue", "Yellow");
        sort($colors);
        $arrlength = count($colors);
        for ($i = 0; $i < $arrlength; $i++) {
            echo $colors[$i];
            echo"<br/>";
        }
        ?>
    </body>
</html>
```

# Example:

- Sorting Associative Arrays in Descending Order By Value

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $age = array("Ali" => "22", "Salma" => "32", "Khalid" => "28");
        arsort($age);
        foreach ($age as $name) {
            echo $name ;
            echo "<br>";
        }
        ?>
    </body>
</html>
```

# Functions

- In addition to the built-in function, PHP also allows us to define our own functions.
- The basic syntax of creating a function can be give with:

```
function functionName( ) {
   code to be executed;
}
```

# Example:

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php

        function helloMsg() {
            echo "Hello world!";
        }

        helloMsg();
        ?>
    </body>
</html>
```

# Functions with Parameters

- Information can be passed to functions through parameters.

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        function getSum($num1, $num2) {
            $sum = $num1 + $num2;
            echo "Sum of the two numbers $num1 and $num2 is : $sum";
        }
        getSum(10, 20);
        ?>
    </body>
</html>
```

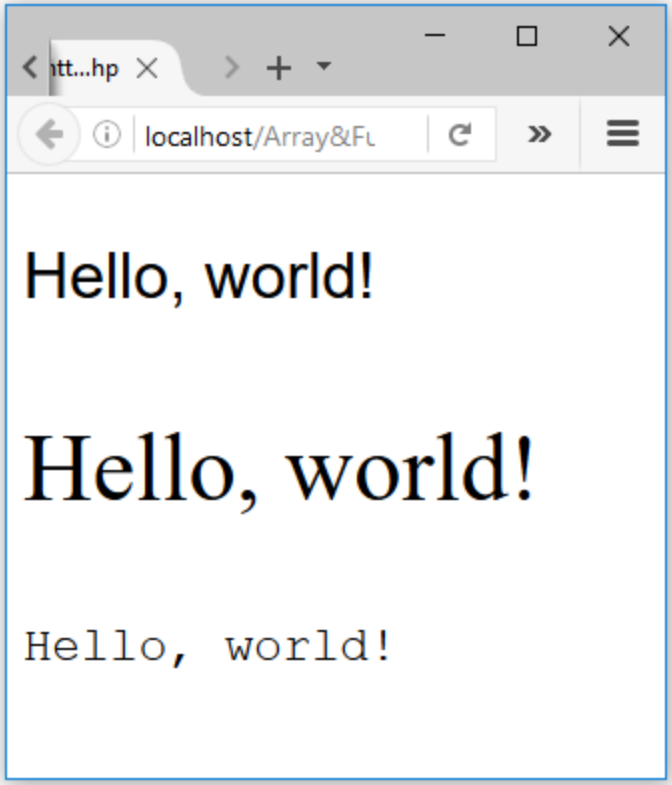
# Functions with Default  Parameters Values

- In PHP, we can create functions with default parameters value

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php

        function customFont($font, $size = 1.5) {
            echo "<p style=\"font-family: $font; font-size: {$size}em;\">Hello, world!</p>";
        }

        customFont("Arial", 2);
        customFont("Times", 3);
        customFont("Courier");
        ?>
    </body>
</html>
```


ttt...hp
localhost/Array&Fu
Hello, world!
Hello, world!
Hello, world!

# Functions Returning Values

- A function can return a value back to the script that called the function using the return statement.
- The value may be of any type, including arrays and objects.

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php

        function getSum($num1, $num2) {
            $total = $num1 + $num2;
            return $total;
        }

        echo "Sum of the two numbers 5 and 10 is :" . getSum(5, 10);
        ?>
    </body>
</html>
```

# Thanks!