

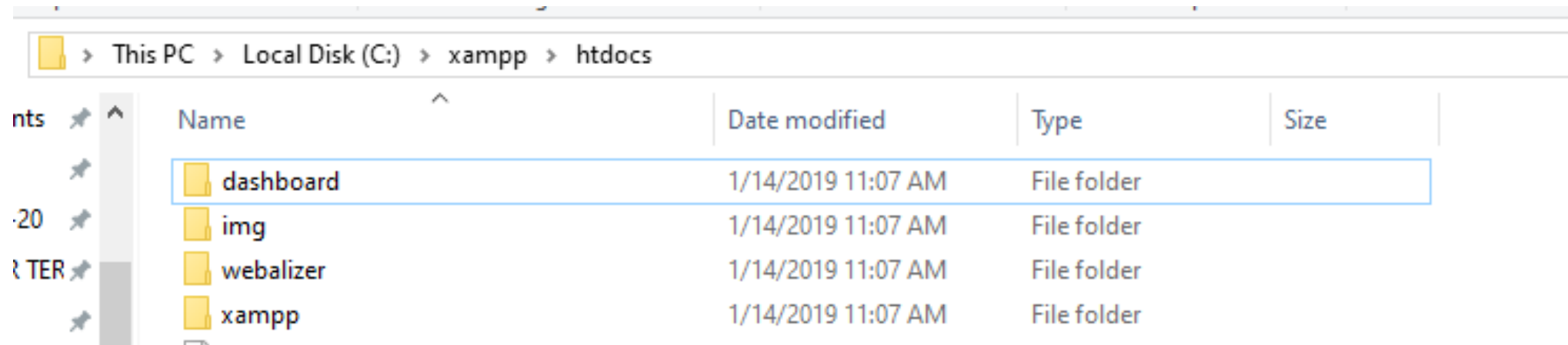
# PHP

## How to write my first php script

# Run Your First PHP Script




Step 1: Go to XAMPP server directory

I'm using Windows, so my root server directory is "C:\xampp\htdocs".



## Step 2: Create hello.php

Create a file and name it "hello.php"

 favicon	7/16/2015 9:02 PM	Icon	31 KB
 hello	7/16/2015 9:02 PM	PHP File	1 KB
 index	7/16/2015 9:02 PM	PHP File	1 KB

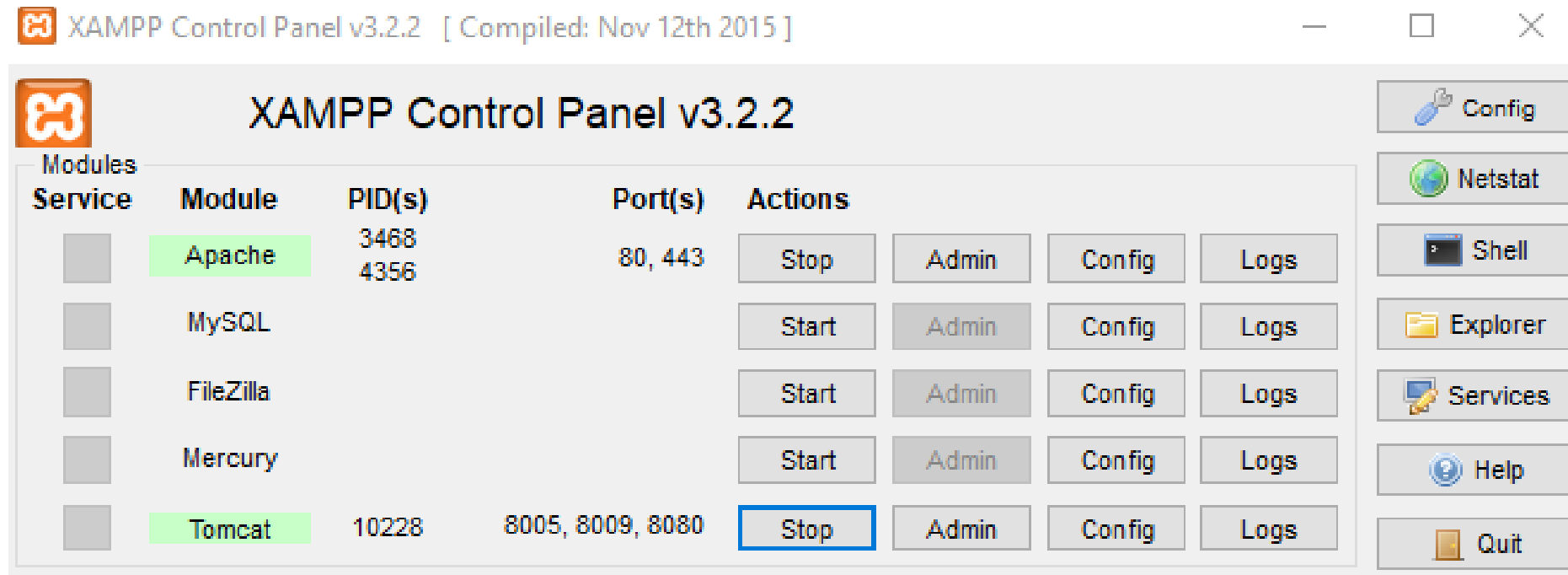
## Step 3: Code Inside hello.php

Open hello.php and put the following code.

```
*hello - Notepad
File Edit Format View Help
<?php
echo "Hello World!";
?>
```

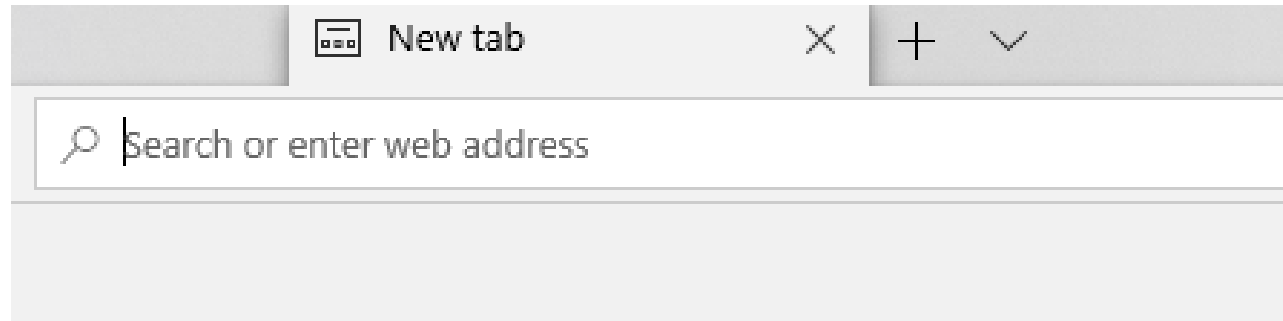
## Step 4: Run your XAMPP server

Open xampp application and click start required module



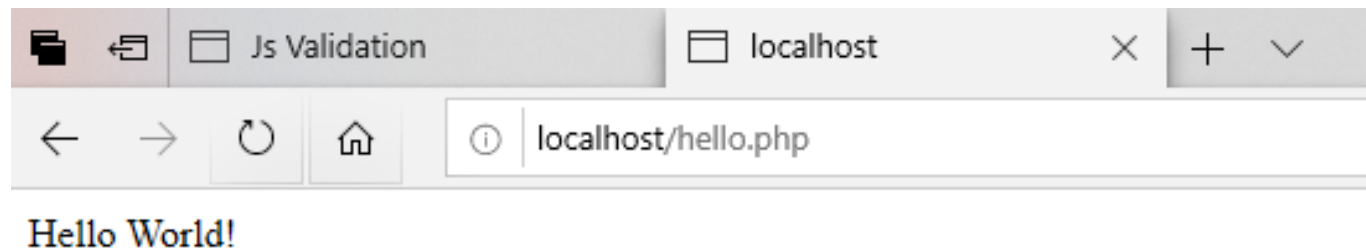
## Step 5: Open New Tab

Run it by opening a new tab in your browser



## Step 6: Load hello.php

On you browser window, type `http://localhost/hello.php`



## "Hello World" Script in PHP

```
<html>
```

```
    <head>
```

```
        <title>Hello World</title>
```

```
    </head>
```

```
    <body>
```

```
        <?php echo "Hello, World!";?>
```

```
    </body>
```

```
</html>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**:

```
<html>
```

```
    <head>
```

```
        <title>Hello World</title>
```

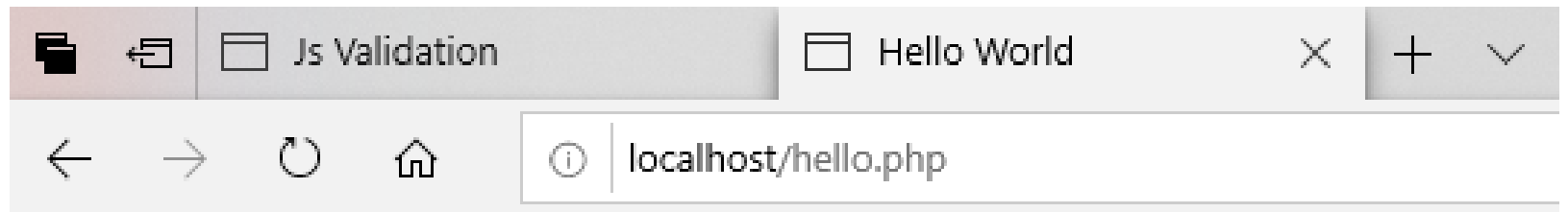
```
    </head>
```

```
    <body>
```

```
        <?php echo "Hello, World!";?>
```

```
    </body>
```

```
</html>
```



Hello, World!

# What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data



# PHP - Environment Setup

Web Server

Database

PHP Parser

## Canonical PHP tags

The most universally effective PHP tag style is –

`<?php...?>`

# Commenting PHP Code

Single-line comments -//

Multi-lines comments -/\*.....\*/

## **PHP is whitespace insensitive**

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row

## **PHP is case sensitive**

Yeah it is true that PHP is a case sensitive language.

Statements are expressions terminated by semicolons

# **Expressions are combinations of tokens**

The smallest building blocks of PHP are the indivisible tokens, such as numbers (3.14159), strings (.two.), variables (\$two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and so forth

## **Braces make blocks**

We can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

# PHP - Variable Types

All variables in PHP are denoted with a leading dollar sign (\$).

The value of a variable is the value of its most recent assignment.

Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.

Variables can, but do not need, to be declared before assignment.

Variables in PHP do not have intrinsic types

# Variable Naming

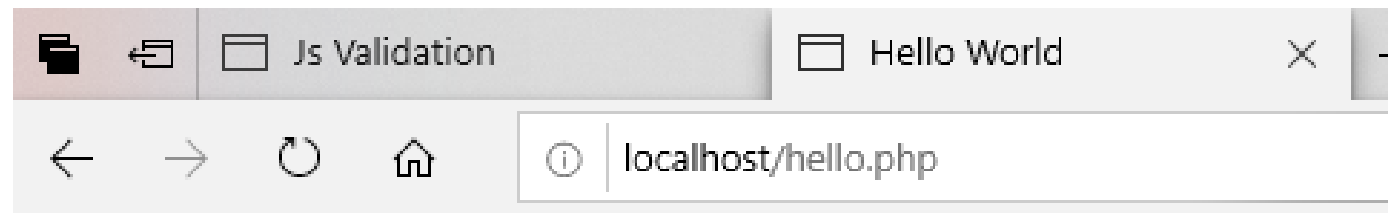
Rules for naming a variable is –

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc

- Integers** – are whole numbers, without a decimal point, like 4195.
- Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- Booleans** – have only two possible values either true or false.
- NULL** – is a special type that only has one value: NULL.
- Strings** – are sequences of characters, like 'PHP supports string operations.'
- Arrays** – are named and indexed collections of other values.
- Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources** – are special variables that hold references to resources external to PHP (such as database connections).

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php
      $x=15;$y=30;
      echo "Sum: ",$x+$y;
    ?>
  </body>
</html>
```



Sum: 45



# Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared.

Local variables

Function parameters

Global variables

Static variables

```
<?php
```

```
$x = 5;
```

```
$y = 10;
```

```
function myTest() {
```

```
    global $x, $y;
```

```
    $y = $x + $y;
```

```
}
```

```
myTest();
```

```
echo $y; // outputs 15
```

```
?>
```

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
```

```
myTest();
myTest();
myTest();
?>
```

Output

0  
1  
2

# PHP - Constants Types

A constant is a name or an identifier for a simple value.

A constant value cannot change during the execution of the script.

By default, a constant is case-sensitive.

By convention, constant identifiers are always uppercase.

A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name.

```
<?php
    define("MINSIZE", 50);

    echo MINSIZE;
    echo constant("MINSIZE"); // same thing as the previous line
?>
```

# **PHP - Operator Types**

Arithmetic Operators

Comparison Operators

Logical (or Relational) Operators

Assignment Operators

Conditional (or ternary) Operators

## Precedence of PHP Operators

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= <small>James Mathew, Precidency University, Dept.CSE</small>	Right to left

Example for practice

```
<?php
    $a = 42;
    $b = 20;
    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";
    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";
    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";
    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";
    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";
    $c = $a++;
```



# PHP - Decision Making

**if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

**elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true

**switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

**Syntax remain same as C language**

Example for practice

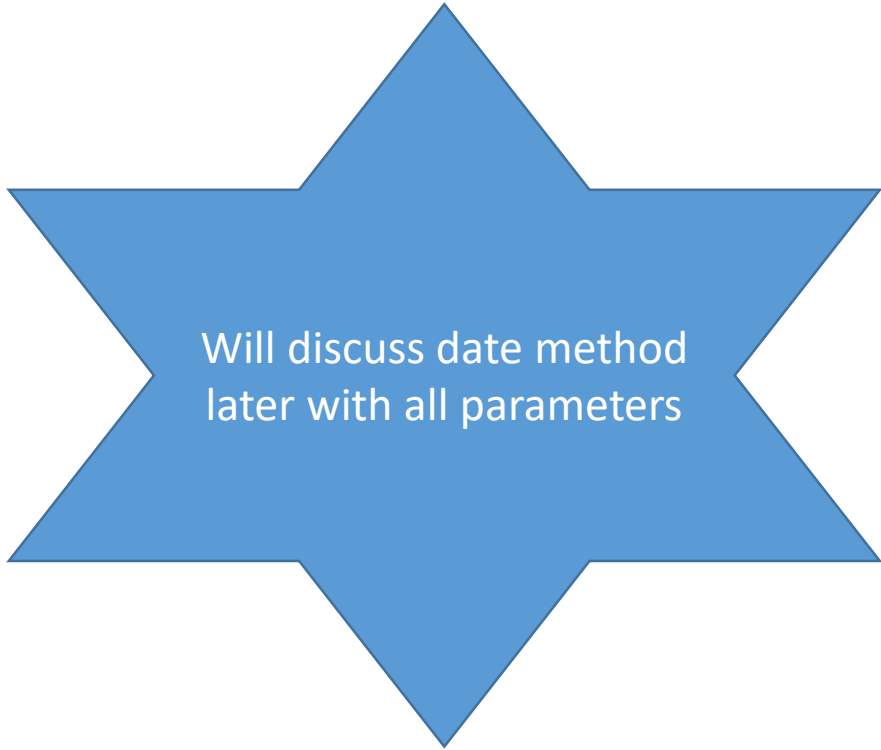
```
<?php
    $d = date("D");//date will return the currrent day

    if ($d == "Fri")
        echo "Have a nice weekend!";

    elseif ($d == "Sun")
        echo "Have a nice Sunday!";

    else
        echo "Have a nice day!";

?>
```



Will discuss date method  
later with all parameters

# PHP - Loop Types

**for** – loops through a block of code a specified number of times.

**while** – loops through a block of code if and as long as a specified condition is true.

**do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.

**foreach** – loops through a block of code for each element in an array.

Example for practice

```
<?php
    $a = 0;
    $b = 0;

    for( $i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
    }

    echo ("At the end of the loop a = $a and b = $b" );
?>
```

```
<html>
<body>

<?php
    $i = 0;
    $num = 50;

    while( $i < 10) {
        $num--;
        $i++;
    }

    echo ("Loop stopped at i = $i and num = $num" );
?>

</body>
</html>
```

```
<html>
<body>

<?php
    $i = 0;
    $num = 0;

    do {
        $i++;
    }

    while( $i < 10 );
    echo ("Loop stopped at i = $i" );
?>

</body>
</html>
```

## Example foreach

```
<html>
```

```
<body>
```

```
<?php
```

```
    $array = array( 1, 2, 3, 4, 5);
```

```
    foreach( $array as $value ) {
```

```
        echo "Value is $value <br />";
```

```
    }
```

```
?>
```

```
</body>
```

```
</html>
```

## OUTPUT

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

## The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely.

## The continue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.

# PHP – Arrays

An array is a data structure that stores one or more similar type of values in a single name.

**Numeric array** – An array with a numeric index. Values are stored and accessed in linear fashion.

**Associative array** – An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.

**Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices



Example for practice. Two methods of using array

```
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}
?>
```

```
<?php
```

```
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";

foreach( $numbers as $value ) {
    echo "Value is $value <br />";
}
?>
```

# Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index.

Associative array will have their index as string so that you can establish a strong association between key and values.

```
$salaries = array("James" => 1000, "Shweta" => 1500, "Sunil" => 2000);
```

```
echo "Salary of james is ". $salaries['James'] .;
```

# String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator

```
<?php
    $string1="Hello World";
    $string2="1234";

    echo $string1 . " " . $string2;
?>
```

Hello World 1234

Example for practice, addition of two numbers using form

```
<body>
```

```
<form>
```

Enter First Number:

```
<input type="number" name="number1" /><br><br>
```

Enter Second Number:

```
<input type="number" name="number2" /><br><br>
```

```
<input type="submit" name="submit" value="Add">
```

```
</form>
```

```
</body>
```

```
<?php
```

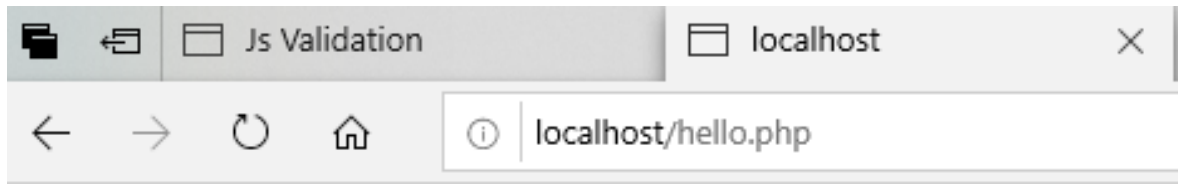
```
$number1=$_GET['number1'];
```

```
$number2=$_GET['number2'];
```

```
$result=$number1+$number2;
```

```
echo "Sum of $number1 and $number2 is=".$result;
```

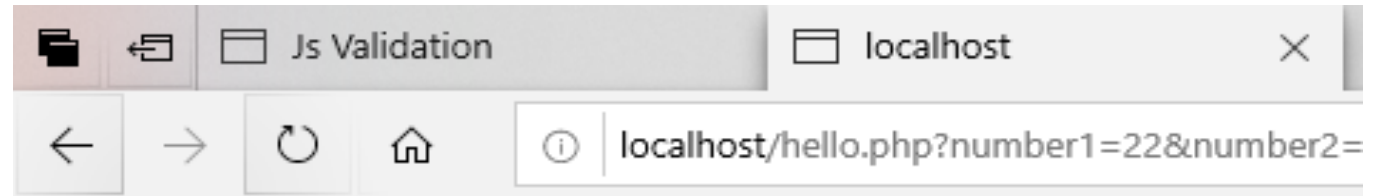
```
?>
```



Enter First Number: 22

Enter Second Number: 10

Add



Enter First Number:

Enter Second Number:

Add

Sum of 22 and 10 is=32

# PHP - GET & POST Methods

There are two ways the browser client can send information to the web server.

The GET Method

The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding.

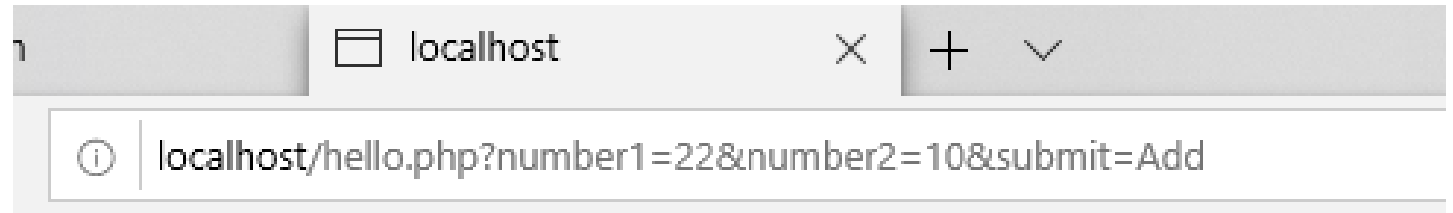
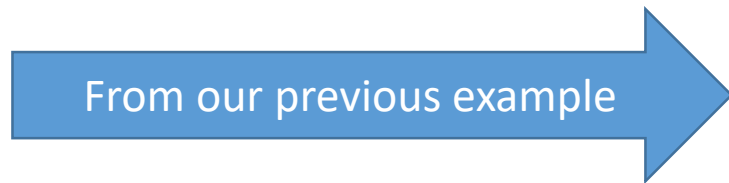
In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

Spaces are removed and replaced with the `+` character and any other nonalphanumeric characters are replaced with a hexadecimal values.

After the information is encoded it is sent to the server.

# The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character.



- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.

Task:-Write a php program to read and display your name age using GET\_METHOD

```
<?php
    if( $_GET["name"] || $_GET["age"] ) {
        echo "Welcome ". $_GET['name']. "<br />";
        echo "You are ". $_GET['age']. " years old.";

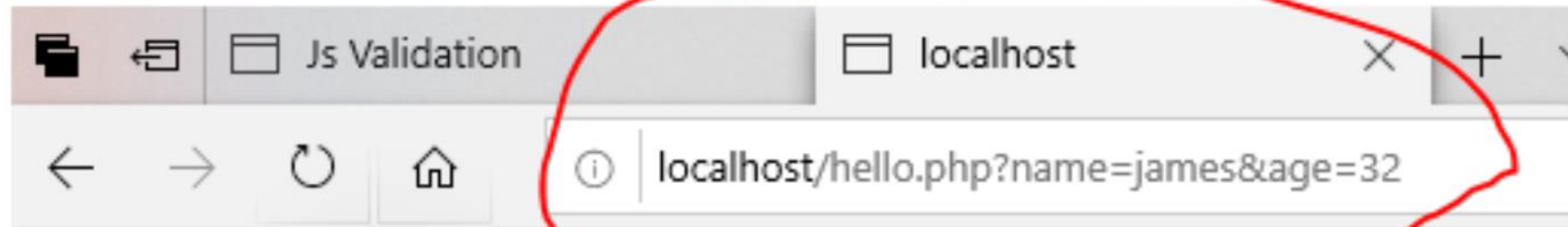
        exit();
    }
?>
<html>
    <body>

        <form action = "<?php $_PHP_SELF ?>" method = "GET">
            Name: <input type = "text" name = "name" />
            Age: <input type = "text" name = "age" />
            <input type = "submit" />
        </form>

    </body>
</html>
```



Name:  Age:



Welcome james  
You are 32 years old.

# The POST Method

The POST method transfers information via HTTP headers.

The POST method does not have any restriction on data size to be sent.

The POST method can be used to send ASCII as well as binary data.

The data sent by POST method goes through HTTP header so security depends on HTTP protocol.

The PHP provides **\$\_POST** associative array to access all the sent information using POST method.

The PHP `$_REQUEST` variable contains the contents of both `$_GET`, `$_POST`

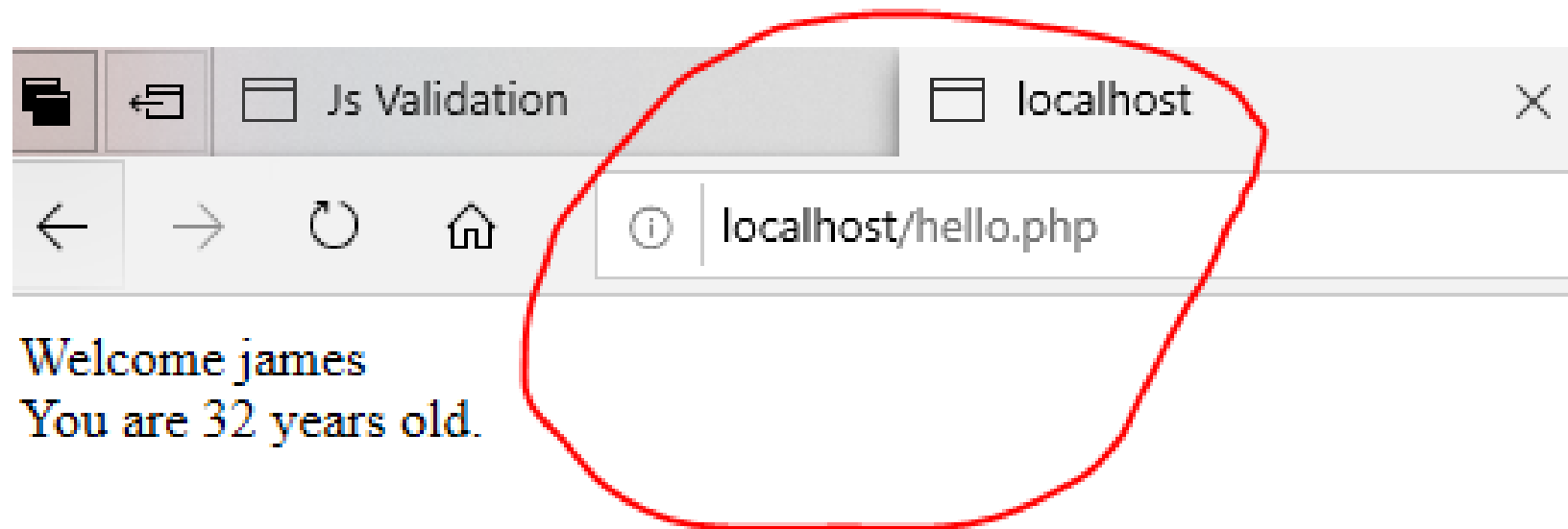
The PHP `$_REQUEST` variable can be used to get the result from form data sent with both the GET and POST methods.

```
<?php
    if( $_REQUEST["name"] || $_REQUEST["age"] ) {
        echo "Welcome ". $_REQUEST['name']. "<br />";
        echo "You are ". $_REQUEST['age']. " years old.";
        exit();
    }
?>
<html>
    <body>

        <form action = "<?php $_PHP_SELF ?>" method = "POST">
            Name: <input type = "text" name = "name" />
            Age: <input type = "text" name = "age" />
            <input type = "submit" />
        </form>

    </body>
</html>
```

Name:  Age:



# PHP - File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it.

The include() Function

The require() Function

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort.

# The include() Function

The include() function takes all the text in a specified file and copies it into the file that uses the include function.

If there is any problem in loading a file then the **include()** function generates a warning but the script will continue execution.

```
<html>
```

```
  <body>
```

```
    <?php include("menu.php"); ?>
```

```
    <p>This is an example to show how to include PHP file!</p>
```

```
  </body>
```

```
</html>
```

# The **require()** Function

The `require()` function takes all the text in a specified file and copies it into the file that uses the `include` function.

If there is any problem in loading a file then the **`require()`** function generates a fatal error and halt the execution of the script.

# PHP – Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

## Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it.



```
<html>
```

```
<head>
```

```
<title>Writing PHP Function</title>
```

```
</head>
```

```
<body>
```

```
<?php
```

```
/* Defining a PHP Function */
```

```
function writeMessage() {
```

```
    echo "You are really a nice person, Have a nice time!";
```

```
}
```

```
/* Calling a PHP Function */
```

```
writeMessage();
```

```
?>
```

```
</body>
```

```
</html>
```

# PHP Functions with Parameters

```
<html>

<head>
  <title>Writing PHP Function with Parameters</title>
</head>

<body>

  <?php
    function addFunction($num1, $num2) {
      $sum = $num1 + $num2;
      echo "Sum of the two numbers is : $sum";
    }

    addFunction(10, 20);
  ?>

</body>
</html>
```

# Passing Arguments by Reference

```
<html>
```

```
<head>
```

```
<title>Passing Argument by Reference</title>
```

```
</head>
```

```
<body>
```

```
<?php
```

```
function addFive($num) {  
    $num += 5;  
}
```

```
function addSix(&$num) {  
    $num += 6;  
}
```

```
$orignum = 10;
```

```
addFive( $orignum );
```

```
echo "Original Value is $orignum<br />";
```

```
addSix( $orignum );
```

```
echo "Original Value is $orignum<br />";
```

```
?>
```

```
</body>
```

```
</html>
```

# PHP Functions returning value

```
<html>

<head>
  <title>Writing PHP Function which returns value</title>
</head>

<body>

  <?php
    function addFunction($num1, $num2) {
      $sum = $num1 + $num2;
      return $sum;
    }
    $return_value = addFunction(10, 20);

    echo "Returned value from the function : $return_value";
  ?>

</body>
</html>
```