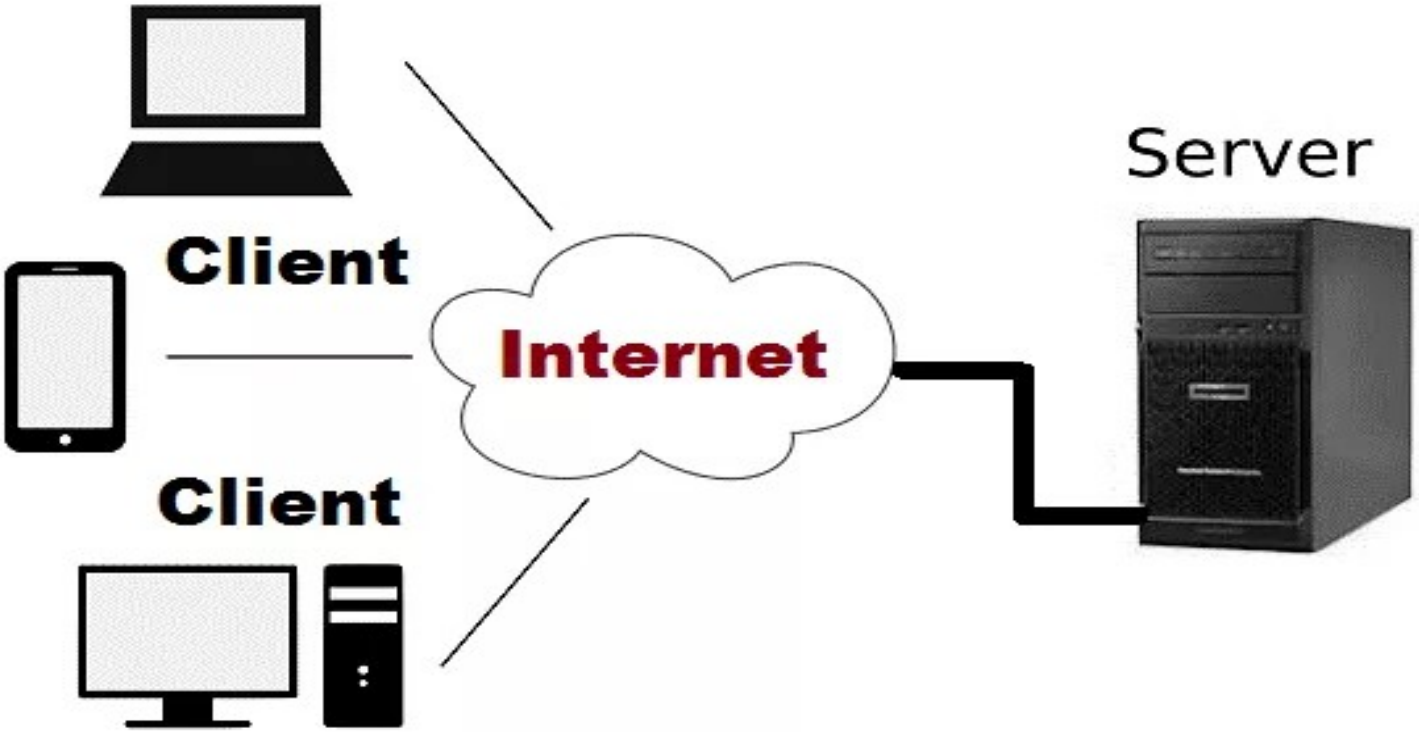
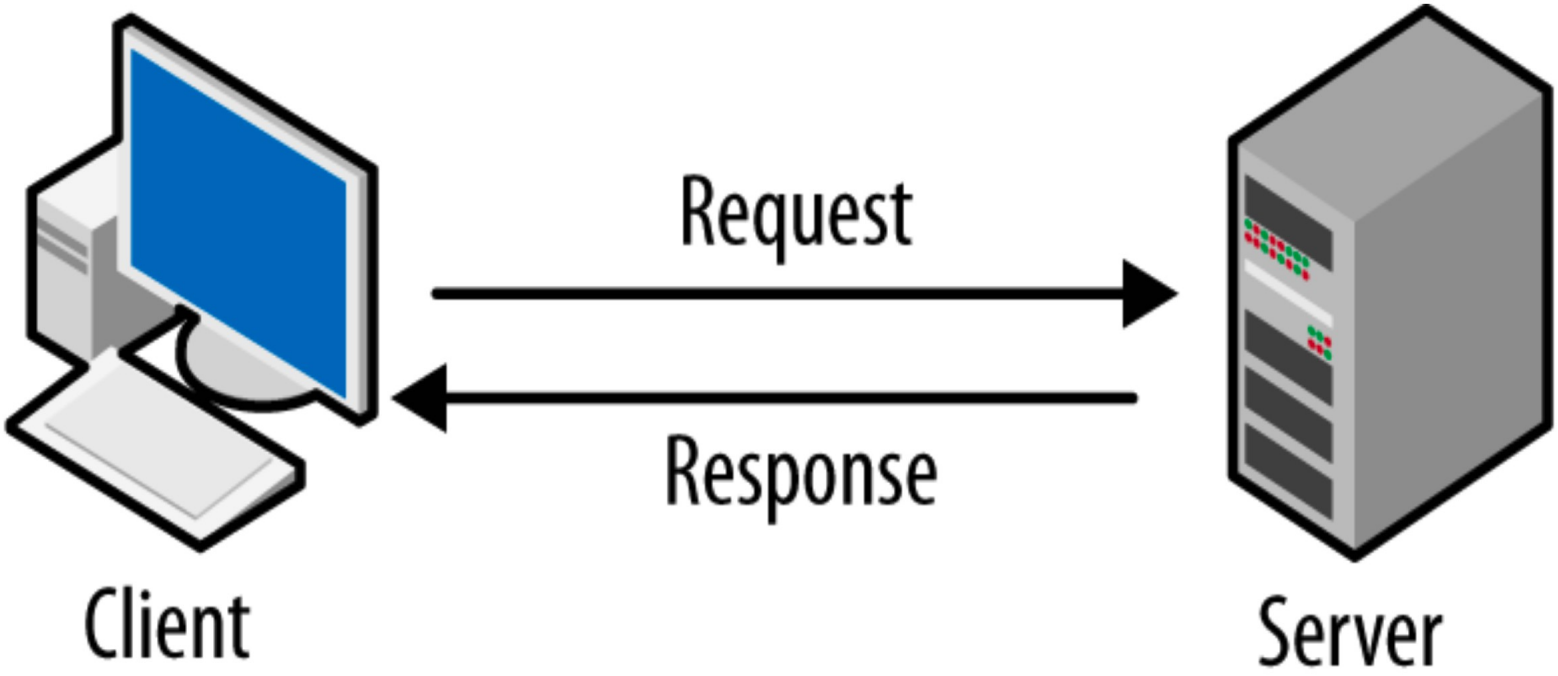
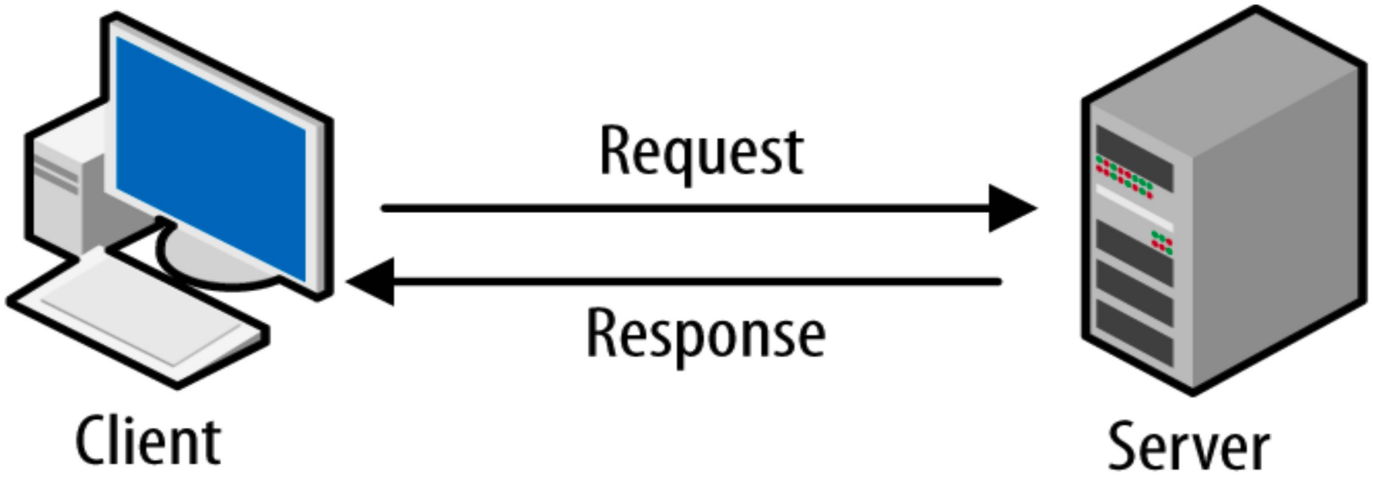
CLIENT-SERVER MODEL



CLIENT-SERVER MODEL



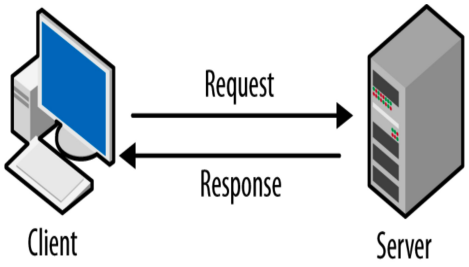
CLIENT-SERVER MODEL : HTTP



CLIENT-SERVER ARCHITECTURE IS AN APPLICATION MODEL THAT DISTRIBUTES TASKS BETWEEN SERVICE PROVIDERS (SERVERS) AND SERVICE DEMAND POINTS (CLIENTS).

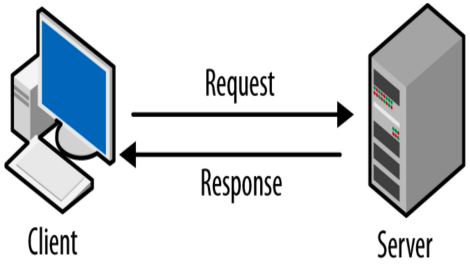
THEORETICALLY, THE SERVER AND THE CLIENT CAN BE ON THE SAME PHYSICAL MACHINE, ALTHOUGH THE MAXIMUM UTILITY IS OBTAINED WHEN THEY ARE TWO DIFFERENT MACHINES

CLIENT-SERVER MODEL



CLIENT CHARACTERISTICS:

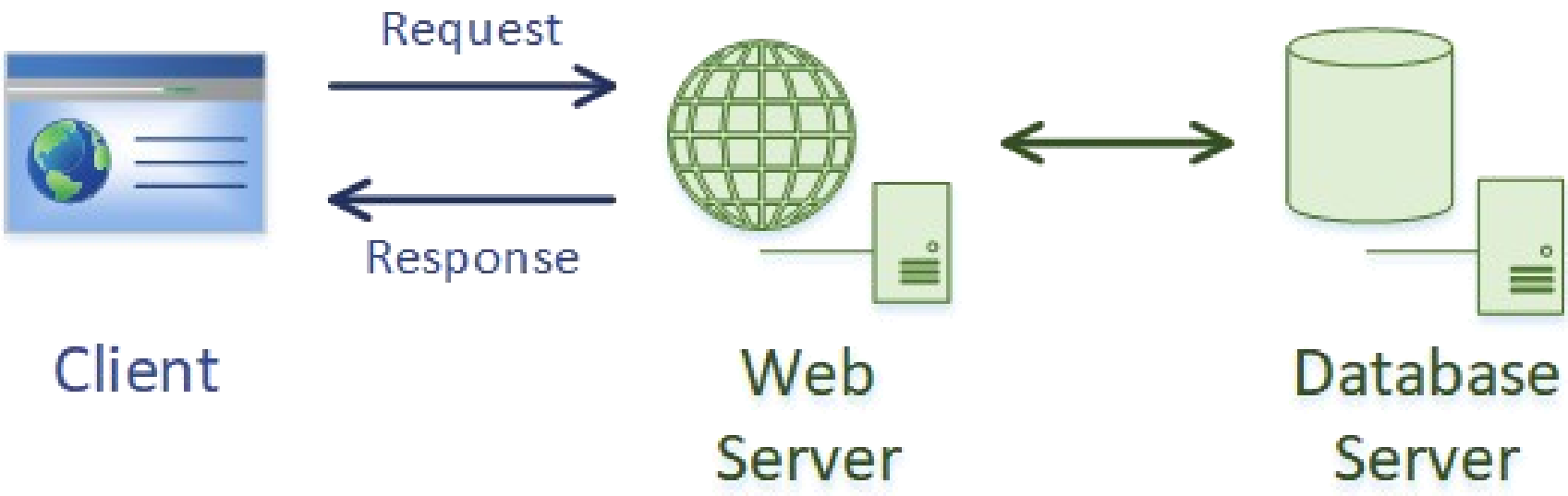
STARTS REQUESTS
WAITS FOR THE SERVER'S RESPONSE
INTERACTS WITH THE USER WITH A GRAPHIC INTERFACE (IN GENERAL)



SERVER CHARACTERISTICS:

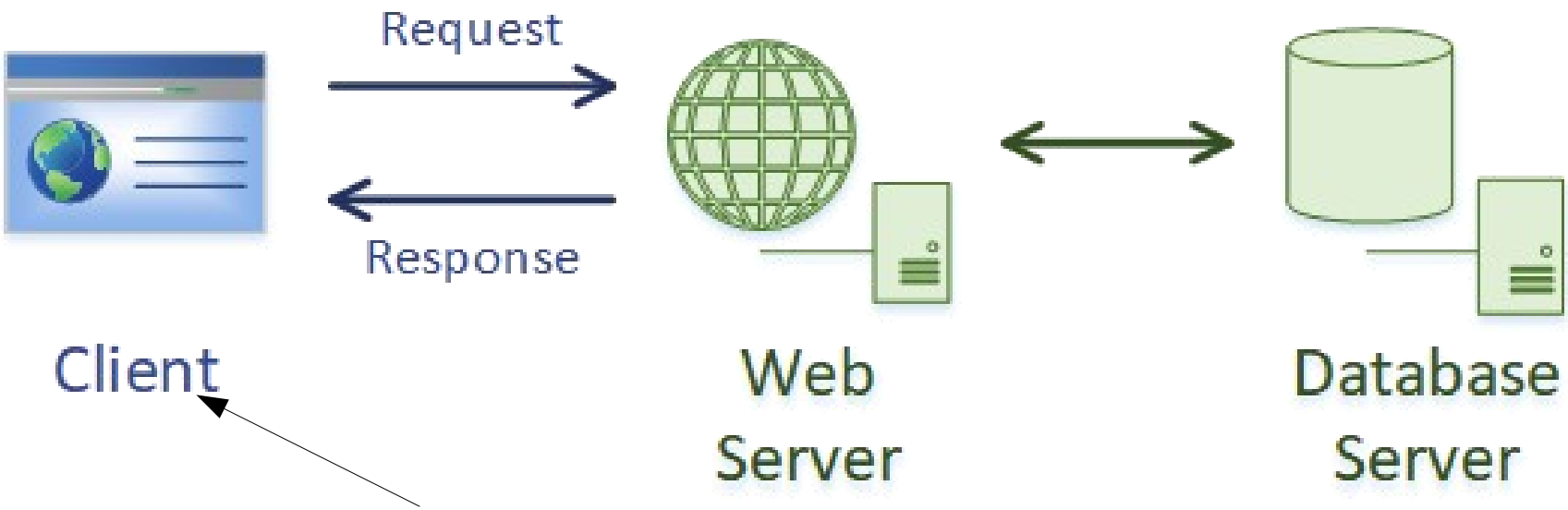
WAITS FOR CLIENT REQUESTS
ATTENDS TO REQUESTS
INTERACTS WITH DATABASES
RETURNS THE RESULT TO THE CLIENT

CLIENT-SERVER MODEL



The idea is to release the client from making requests, processing and manipulating information in most cases. The server will do all the processing and generates results in a format that the client can understand.

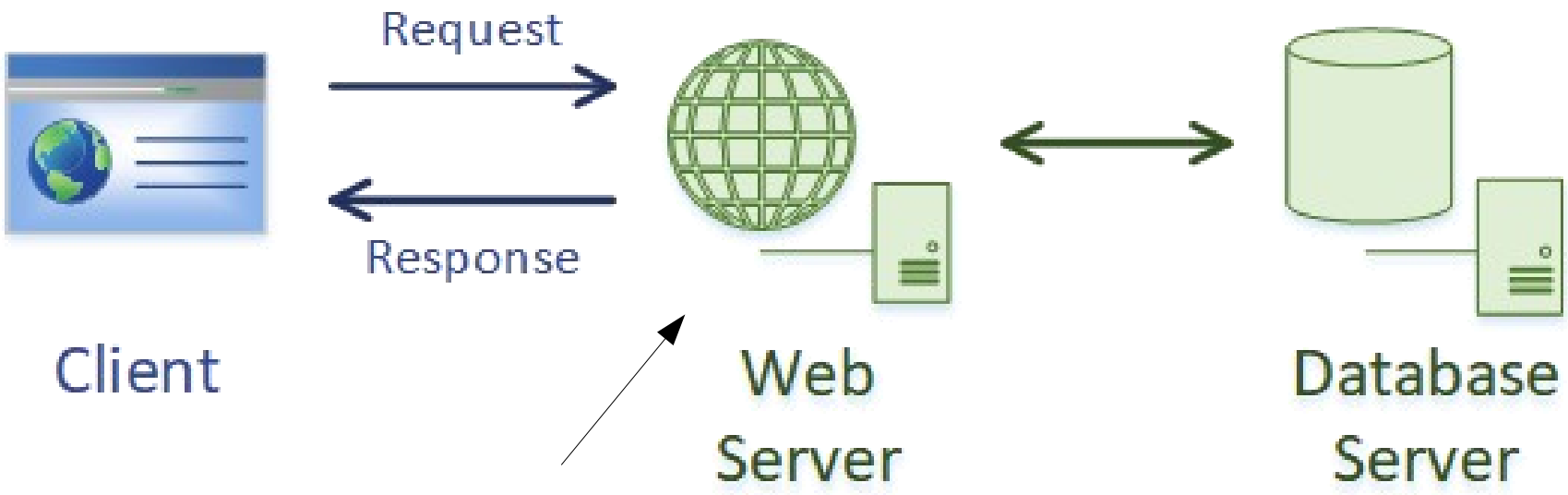
CLIENT-SERVER MODEL



The languages that the client understands and can interpret are CLIENT LANGUAGES

They are used to program scripts or small programs that facilitate a certain interactivity (menus, image movement, format control ...). They do not generate any type of compiled code, they are incorporated into the HTML code so that they are directly recognized and interpreted by the web browser.

CLIENT-SERVER MODEL



The languages that the server understands and can interpret are SERVER LANGUAGES

The generated programs are stored on the server, which will be responsible for running them and sending to the clients the information resulting from the execution, in a format (usually HTML) We will not be able to see the source code of the programs. The client's workload decreases, and the server's workload increases

CLIENT LANGUAGES

HTML
CSS
JavaScript
Java (Applets)

SERVER LANGUAGES

PHP
ASP
C++
Python (Django, Flask)
Java (JSP / Servlets)
JavaScript (NodeJS)

CLIENT-SERVER MODEL ADVANTAGES

- Centralization: if one client turn off, everything keeps working
- Maintenance and scalability: change, add, remove or modify machines
- Security: transaction control, data encapsulation
- Optimization: Client programs are simple and do not require too many resources

CLIENT-SERVER MODEL DISADVANTAGES

- Centralization: if the server turns off the application stops working
- Traffic: large number of communications (requests and responses) simultaneously
- Cost: Server machines need specific software and hardware

STATIC AND DYNAMICS PAGES

Static: Static web pages contain information that doesn't change until the web designer or programmer manually changes it. It changes infrequently. Designed primarily using the HTML language.

Dynamics: They make it possible for the user to change their appearance and even their content. Dynamic web pages allow you to easily change your content in real time without even touching the coding of the page.

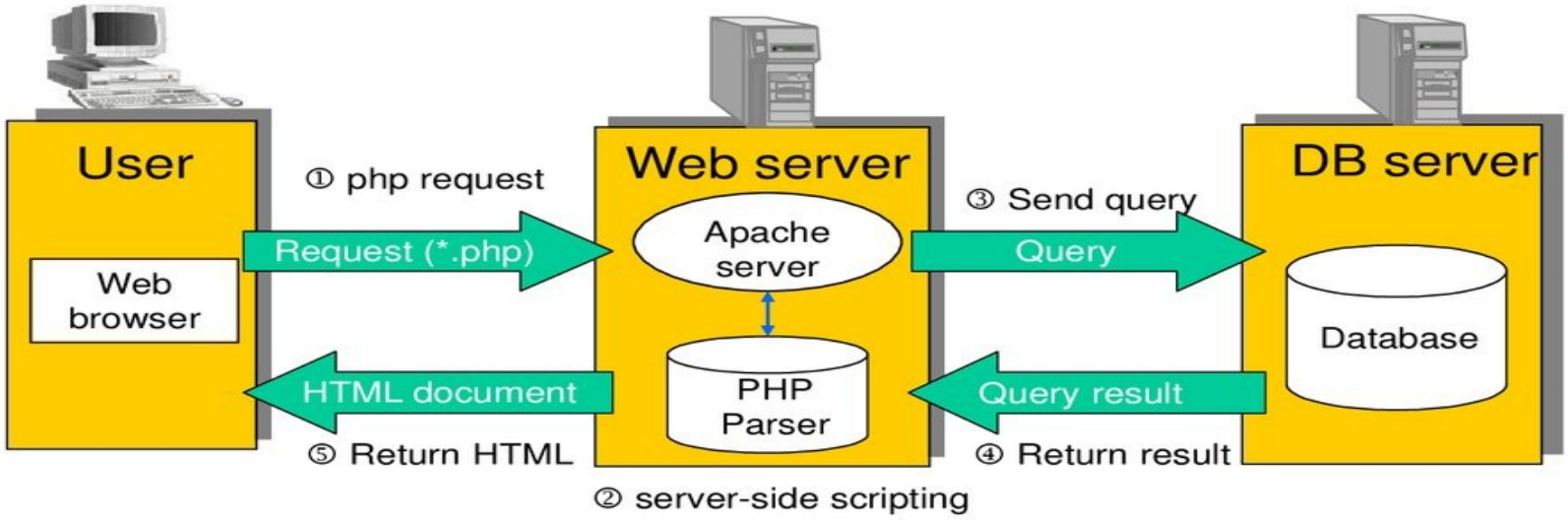
You have to know a method for automatically inserting real-time data into the HTML code. This is where web scripting languages come in.

CLIENT-SERVER MODEL WITH PHP

We will use PHP as the web language. We will install a web server (Apache)

PHP

- Architecture Overview



CS360, KAIST 12

CLIENT-SERVER MODEL WITH PHP

ALLOWS THE GENERATION OF DYNAMIC AND INTERACTIVE WEB PAGES, WHERE THE RESULT OBTAINED BY THE CLIENT DEPENDS ON THE REQUEST MADE TO THE SERVER

SERVER AND MARKUP LANGUAGE INTEGRATION

index.php

Source History

```
1  <!DOCTYPE html>
2  <!--
3  To change this license header, choose License Headers in Project Properties.
4  To change this template file, choose Tools | Templates
5  and open the template in the editor.
6  -->
7  <html>
8      <head>
9          <meta charset="UTF-8">
10         <title></title>
11     </head>
12     <body>
13         <?php
14         // put your code here
15         ?>
16     </body>
17 </html>
18
```

PHP AND HTML

To start developing in PHP, create a plain text file with a .php file extension and open it with the editor of your choice –for example VSCode, Sublime, Notepad, jEdit, Dreamweaver, NetBeans, or Eclipse PDT

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP Test</title>
    </head>
    <body>
    </body>
</html>
```

PHP AND HTML

PHP code can be embedded anywhere in a web document in several different ways. In the standard notation we delimite the code by <? php and ?>. This is called a PHP code block, or just a PHP block.

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP Test</title>
    </head>
    <body>
    <p>My first page
        <?php
        echo "Hello, World";
        ?>
    </p>
</body>
</html>
```

PHP AND HTML

Comments are used to insert notes into the code. They have no effect on the parsing of the script. PHP has two standard notations for single-line (//) and multiline (/* */) comments.

The Perl comment notation (#) may also be used to make single-line comments.

```
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
      <?php
         // single-line comment
        # single-line comment
         /* multi-line
         comment */
      ?>
    </body>
</html>
```

VARIABLES

```
<body>
  <?php
      $variable=1;                    // numeric variable
      $cad="String";                //  String variable
      $tab1=array("a","b","c");   // array
      $tab2=["a","b","c"];       // array up to version 5
  ?>
</body>
```

A variable starts with a dollar sign ($) followed by an identifier, which is the name of the variable.

PHP is a loosely typed language. This means that the type of data that a variable can store is not specified

PHP strings can be delimited in four different ways. There are two common notations: double quote (" ") and single quote (' ').

The instructions end with a semicolon (;)

VARIABLE NAMES

Keep in mind that variable names are case sensitive.

They can include underscore, characters and numbers.

But they cannot start with a number.

They also cannot contain spaces or special characters, and they must not be a reserved keyword.

ARITHMETIC OPERATORS

The arithmetic operators include the four basic arithmetic operations, as well as the modulus operator (%), which is used to obtain the division remainder.

```
$x = 4 + 2; // 6 // addition
$x = 4 - 2; // 2 // subtraction
$x = 4 * 2; // 8 // multiplication
$x = 4 / 2; // 2 // division
$x = 4 % 2; // 0 // modulus (division remainder)
```

An exponentiation operator (**) was introduced in PHP 5.6. It raises the left-side operand to the power of the right-side operand.

```
$x = 4 ** 2; // 16 // exponentiation
```

We can use the pow() function as well

COMPARISON OPERATORS

The comparison operators compare two values and return either true or false. They are mainly used to specify conditions, which are expressions that evaluate to either true or false.

```
$x = (2 == 3); // false // equal to
$x = (2 != 3); // true // not equal to
$x = (2 <> 3); // true // not equal to (alternative)
$x = (2 === 3); // false // identical (type and content)
$x = (2 !== 3); // true // not identical
$x = (2 > 3); // false // greater than
$x = (2 < 3); // true // less than
$x = (2 >= 3); // false // greater than or equal to
$x = (2 <= 3); // true // less than or equal to
```

The strict equality operators, `===` and `!==`, are used for comparing both type and value.

Combined Assignment Operators

A common use of the assignment and arithmetic operators is to operate on a variable and then to save the result back into that same variable. These operations can be shortened with the combined assignment operators.

```
$x += 5; // $x = $x+5;
$x -= 5; // $x = $x-5;
$x *= 5; // $x = $x*5;
$x /= 5; // $x = $x/5;
$x %= 5; // $x = $x%5;
$x **= 5; // $x = $x**5;
```

Increment and Decrement Operators

Another common operation is to increment or decrement a variable by one. This can be simplified with the increment (++) and decrement (--) operators.

```
$x++; // $x += 1;
$x--; // $x -= 1;
```

String Concatenation

PHP has two string operators. The dot symbol is known as the concatenation operator (.). It combines two strings into one.

```
$a = 'Hello';
$b = $a . ' World'; // Hello World
```

It also has an accompanying assignment operator (.=), which appends the right-hand string to the left-hand string variable.

```
$a .= ' World'; // Hello World
```

You can combine types:

```
$a=1;
$b=2;
echo "a variable is ".$a." and b variable is ".$b;
```

LOGICAL OPERATORS

and - &&: The logical operators are often used together with the comparison operators. Logical and (&&)evaluates to true if both the left and right side are true

or - ||: evaluates to true if either the left or right side is true.

!: logical not

```
$x = (true && false); // false // logical and
$x = (true || false); // true // logical or
$x = !(true); // false // logical not
```

HTML TAGS IN PHP

```
<body>
  <?php
      $a=1;
      $b=2;
      echo "A Variable is ".$a."<br/>";
      echo "and b Variable is ".$b."<br/>";
  ?>
</body>
```

We can put HTML characters and expressions inside PHP strings, and the browser will interpret them correctly

DOUBLE QUOTE (" ") AND SINGLE QUOTE (' ')

```
<body>
  <?php
        $name="John";
        echo "Hello, $name <br/>"; // Hello, John
        echo 'Hello, $name <br/>'; // Hello, $name
  ?>
</body>
```

Variables within double quotes are changed by their value
Variables within single quotes are not changed by their value, but you can also use it

STRING VARIABLES

There are two more notations: heredoc and nowdoc. These notations are mainly used to include larger blocks of text.

heredoc syntax consists of the <<< operator followed by an identifier and a new line. The string is then included followed by a new line containing the identifier to close the string. Variables are parsed inside of a heredoc string, just as with double-quoted strings.

```
$s = <<<LABEL
Heredoc (with parsing)
LABEL;
```

nowdoc sintax: same sintax, except that the initial identifier is enclosed in single quotes. Variables are not parsed inside a nowdoc string

```
$s = <<<'LABEL'
Nowdoc (without parsing)
LABEL;
```

OUTPUTTING TEXT: ECHO / PRINTF

```
<?php
    $name="John";
    $age=33;
    echo "Hello, $name, you are $age years old <br/> ";
    printf("Hello, %20s, you are %3d years old<br/>", $name,
  $age);
    $newstring=sprintf("Hello, %20s, you are %3d years old<br/>",
  $name, $age);
    echo $newstring;
?>
```

echo: Outputs one or more expressions, with no additional newlines or spaces.

printf(): Outputs a formatted string (like java)

sprintf(): the same as printf but returns a formatted string