Introduction to PHP

Web Development and Security (ZEIT3119)

Week 6

Dr. Reza Rafeh

Revision

slido

Join at

slido.com

#3986 139

Outline

Web Programming Languages

- PHP
- ASP
- Perl
- Java
- Python

Introduction to PHP

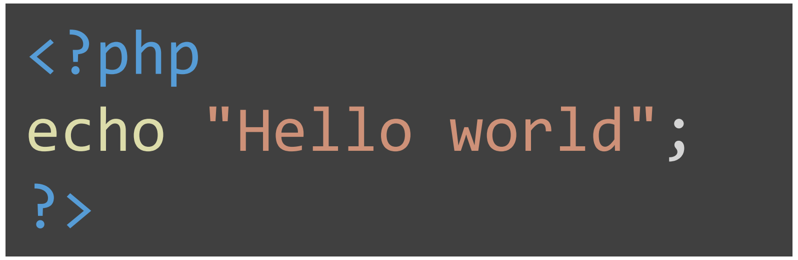
- PHP stands for PHP: Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- Roots are from C and Perl, but very similar to Java
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc..)
- PHP is an open source software (OSS)
- PHP is free to download and use
- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What can PHP do?

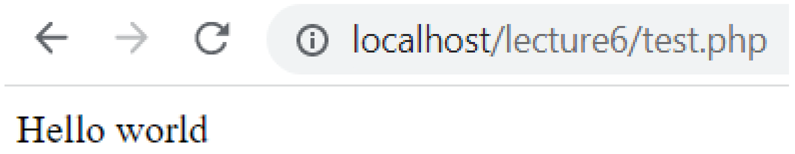
- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- It is embedded in HTML files but all the PHP tags are replaced by the server before anything is sent to the web browser

What can PHP do?

- PHP tags
- <?php ……………… ?>
- Example



```
<?php
echo "Hello world";
?>
```

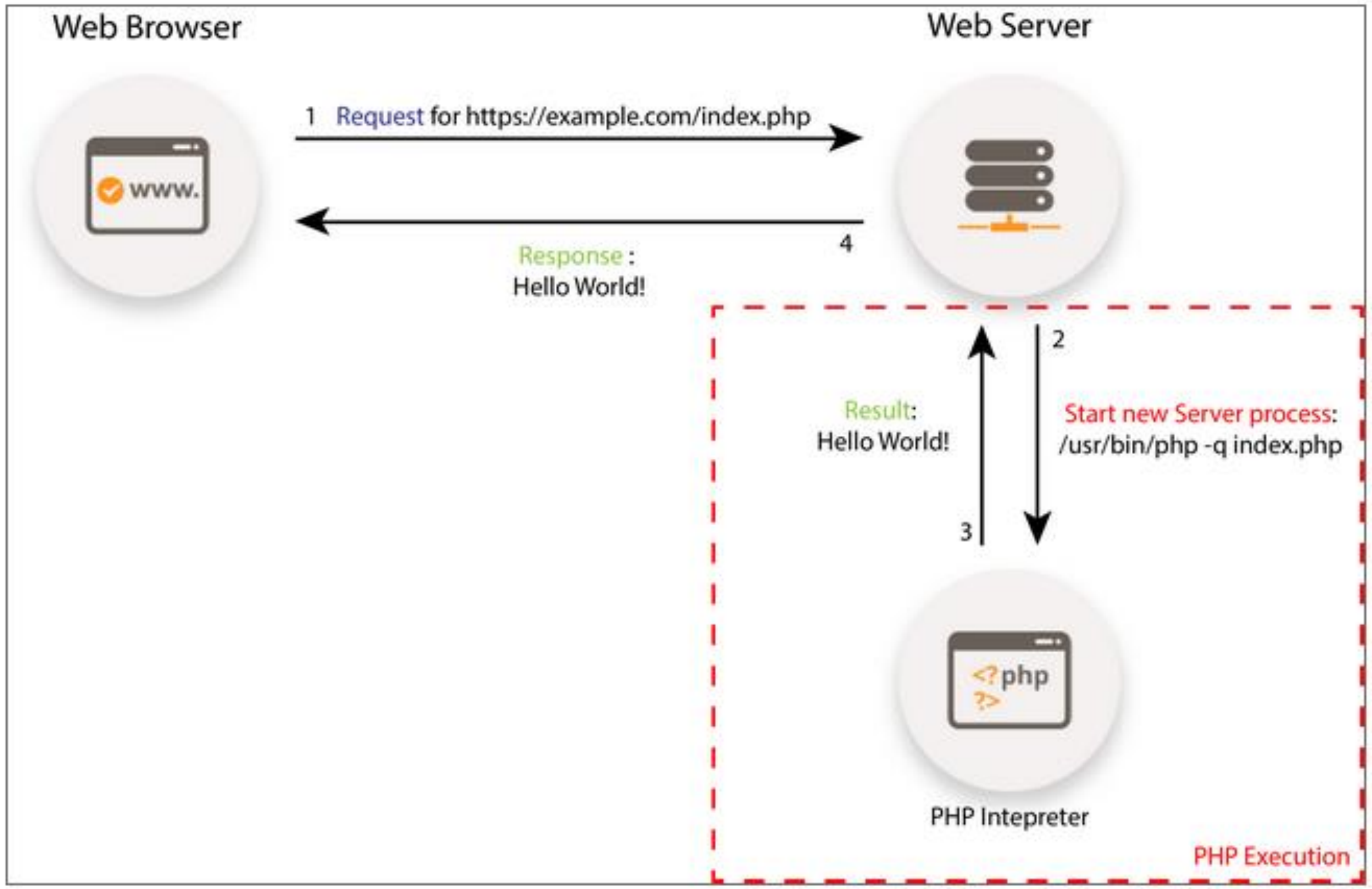


localhost/lecture6/test.php

```
Hello world
```

- “echo” is a built-in function to output the text. Echo and print are the same, only difference is print returns value 1 whereas echo returns nothing. Echo is faster than print and uses different expression
- Common error is forgetting the semi colon

PHP Execution



PHP – Working with Constants

- To define a constant you have to use **define()** function
- To retrieve the value of a constant, you have to simply specify its name.
- A defined constant can never be changed or undefined.
- There is no need to have a constant with a $.
- A valid constant name starts with a letter or underscore.
- The constant() function will return the value of the constant, but you do not know its name, i.e., it is stored in a variable or returned by a function.
- Example

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

Instead of echo MINSIZE use
echo constant("MINSIZE"); // gives the same result
```

PHP – Working with Variables

- Variables are used in storing values, such as numbers, strings or function results
- A variable must start with a letter or underscore. It cannot start with a number
- Variable names are case-sensitive.
- Two words in a variable cannot have spaces but there should be a underscore between them
- $ is used in front of all variables
- It helps to make PHP parser faster
- Example

  $x += 10; // Increment $x by 10
- // indicates comments
- /* */ is used for multiline comment, mostly at t
- isset() function checks if a variable is empty

```php
<?php
$mycounter = 1;
$mystring = "Hello";
$myarray = array("One", "Two", "Three");
echo $myarray[1]; // Prints Two
if (isset($mycounter)) {
    echo "Variable 'mycounter' is set.<br>";
  }
?>
```

PHP – Working with Strings

- A string variable is used to store and manipulate a piece of text
- Each string should be written in either quotation marks (“”) or single quotes (‘’)
- Example

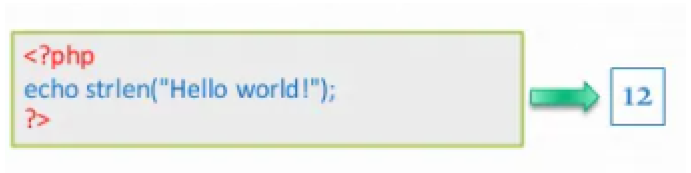
```
<?php
$username = "Fred Smith";
echo $username;
echo "<br>";
$current_user = $username;
echo $current_user;
?>
```

- Concatenation operator (.) is used to put two string vales together

```
<?php
$txt1=“Hello”;
$txt2=“How are you?”;
Echo$txt1. “ ”.$txt2;
?>
```

PHP – Working with Strings

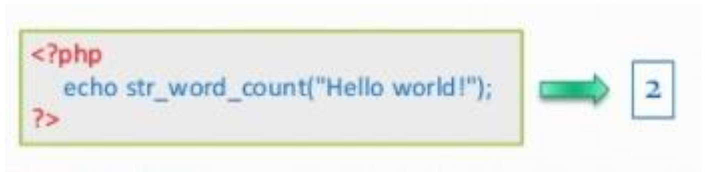
- strlen() function is use to find the length of the string



```
<?php
echo strlen("Hello world!");
?>
```

```
12
```

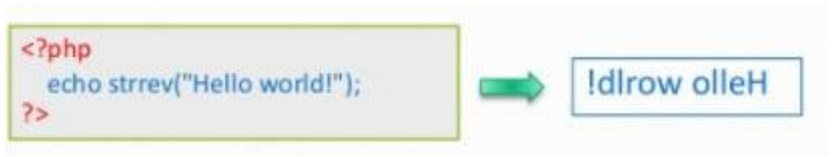
- st_word_count() function is use to count the number of words in the string



```
<?php
   echo str_word_count("Hello world!");
?>
```

```
2
```

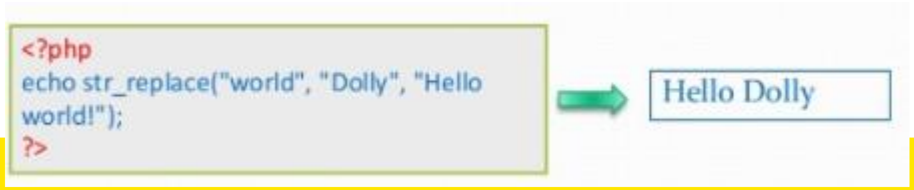
- strrev() function reverses the string



```
<?php
  echo strrev("Hello world!");
?>
```

```
!dlrow olleH
```

- str_replace() function replaces characters with some other characters in a string

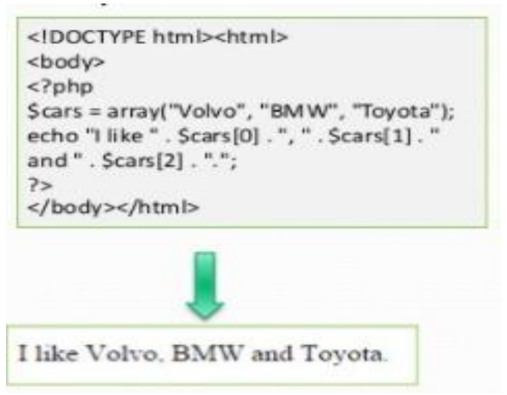


```
<?php
echo str_replace("world", "Dolly", "Hello
world!");
?>
```

```
Hello Dolly
```

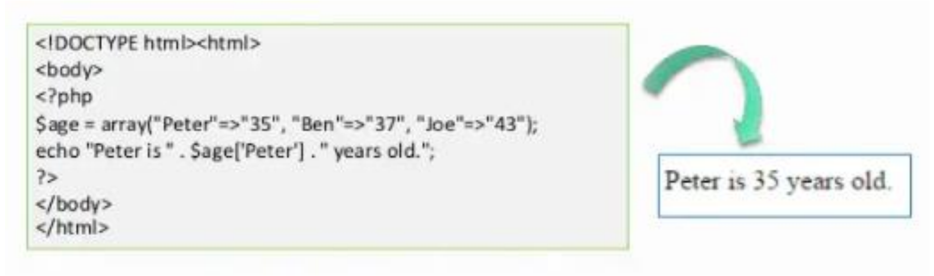
PHP- Array

- Array: displaying or arranging things in a particular way
- It is a variable which can hold more than 1 value at a time
- Stores multiple values in a single variable
- In PHP we have three different types of array
- **Indexed Arrays** – Array with numeric index
- **Associative Arrays** – Arrays with named keys



```
<!DOCTYPE html><html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . "
and " . $cars[2] . ".";
?>
</body></html>
```

```
I like Volvo, BMW and Toyota.
```

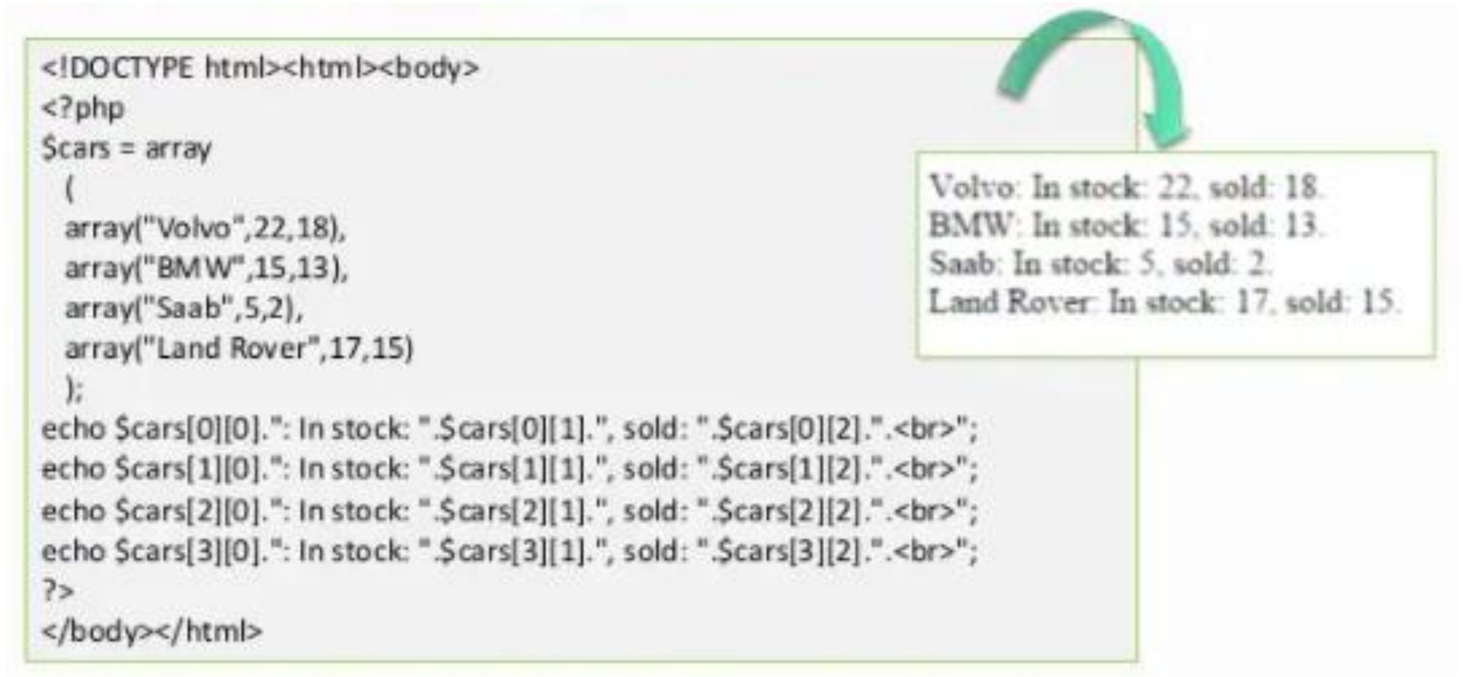


```
<!DOCTYPE html><html>
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
</body>
</html>
```

```
Peter is 35 years old.
```

PHP- Array

- **Multidimensional Arrays** – Array containing 1 or more arrays



```
<!DOCTYPE html><html><body>
<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
</body></html>
```

```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

Commonly Used Array Functions

- array_combine --Creates an array by using one array for keys and another for its values
- array_count_values --Counts all the values of an array
- Array_diff -Computes the difference of arrays
- Array_merge -Merge one or more arrays
- Array_merge_recursive -Merge two or more arrays recursively
- Array_reverse -Return an array with elements in reverse order
- Array_search -Searches the array for a given value and returns the corresponding key if successful
- Array_sum -Calculate the sum of values in an array
- Arsort-Sort an array in reverse order and maintain index association
- Asort-Sort an array and maintain index association
- Krsort-Sort an array by key in reverse order
- Ksort-Sort an array by key

Looping

- ➢ Loops provide a way to repeat commands and control how many times they are repeated.

For loop

- ➢ is usually controlled by counting
- ➢ There is an index variable that you increment or decrement each time through the loop
- ➢ When the index reaches some limit condition, then the looping is done and the code continues

While loop

- ➢ A while statement is like a repeating if statement. Like an If statement, if the test condition is true: the statements get executed.
- ➢ The difference is that after the statements have been executed, the test condition is checked again.
- ➢ If it is still true the statements get executed again. This cycle repeats until the test condition evaluates to false.

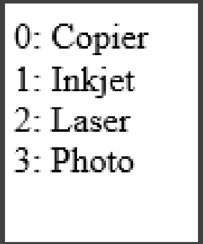
Looping

When the loop is required?

- Do something for a given number of times or for every object in a collection of objects
- For every radio button in a form, see if it is checked
- For every month of the year, charge $100 against the balance
- Calculate the sum of all the numbers in a list
- Many loops are counting loops
- They do something a certain number of times

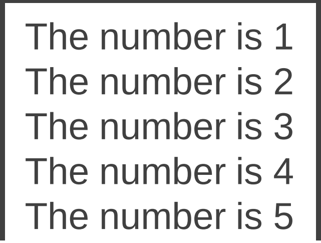
PHP Array and Loop - Examples

```php
<?php
$paper = array("Copier", "Inkjet", "Laser",
"Photo");
$j = 0;
Foreach($paper as $item)
{
echo "$j: $item<br>";
++$j;
}
?>
```



```
0: Copier
1: Inkjet
2: Laser
3: Photo
```

```php
<html>
    <body>
        <?php
            $i=0;
            do {
                $i++;
                echo "The number is " . $i . "<br/>";
            }
            while ($i<5);
        ?>
</body>
</html>
```



```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

If & Else Statements

- The if, elseif and else statements in PHP are used to perform different actions based on different conditions.
- Conditional Statements
- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
- 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 several condition are true

If & Else Statements - Examples

```
<html>
    <body>
        <?php
            $d=date("D");
            if ($d=="Fri"){
                echo "Hello!<br/>";
                echo "Have a nice weekend!";
            }
            else
                echo "See you on Monday!";
        ?>
    </body>
  </html>
```

```
<html>
    <body>
        <?php
            $d=date("D");
            if ($d=="Fri")
                echo "Have a nice weekend!";
            elseif($d=="Sun")
                    echo "Have a nice Sunday!";
                else
                    echo "Have a nice day!";
        ?>
    </body>
</html>
```

Switch Statements

- Similar to if statements but try to avoid long blocks of if..elseif..else code
- If you want to select one of many blocks of code to be executed use the Switch statement

```
<html>
    <body>
        <?php
            $x = 17;
            switch ($x) {
                case 1:
                    echo "Number 1";
                    break;
                case 2:
                    echo "Number 2";
                    break;
                case 3:
                    echo "Number 3";
                    break;
                default:
                    echo "No number between 1 and 3";
            }
        ?>
    </body>
</html>
```

Functions in PHP

- ➢ The real power of PHP comes from its functions.
- ➢ In PHP -there are more than 700 built-in functions available.
- ➢ Create a PHP Function
- ➢ A function is a block of code that can be executed whenever we need it.
- ➢ Creating PHP functions:

- All functions start with the word "function()"
- Name the function -It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{"-The function code starts after the opening curly brace
- Insert the function code
- Add a "}"-The function is finished by a closing curly brace
- One or more parameters, separated by commas, are optional (as indicated by the square brackets).

Functions in PHP

```php
<?php
echo strrev(" .dlrow olleH"); // Reverse string
echo str_repeat("Hip ", 2); // Repeat string
echo strtoupper("hooray!"); // String to
uppercase
?>
```

```
Hello world. Hip Hip HOORAY!
```

```php
<?php
function fix_names($n1, $n2, $n3)
{
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
    return $n1 . " " . $n2 . " " . $n3;
}
echo fix_names("WILLIAM", "henry", "gatES");
?>
```

```
William Henry Gates
```

Use of return

```php
<?php
Function square($num){
    return $num * $num;
}
echo square(4);
?>
```

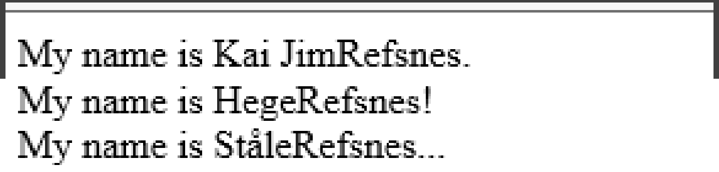
Answer: 16

Function - Example

Adding Parameters

- writeMyName() - is a very simple function. It only writes a static string
- To add more functionality to a function, we can add parameters.
- A parameter is just like a variable.
- You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.
- The following example will write different first names, but the same last name

```
<html>
    <body>
        <?php
        function         writeMyName($fname,$punctuation) {
            echo $fname. "Refsnes" . $punctuation . "<br/>";
        }
        echo "My name is ";
        writeMyName("Kai Jim ",".");
        echo "My name is ";
        writeMyName("Hege ","!");
        echo "My name is ";
        writeMyName("Ståle ","...");
        ?>
    </body>
</html>
```

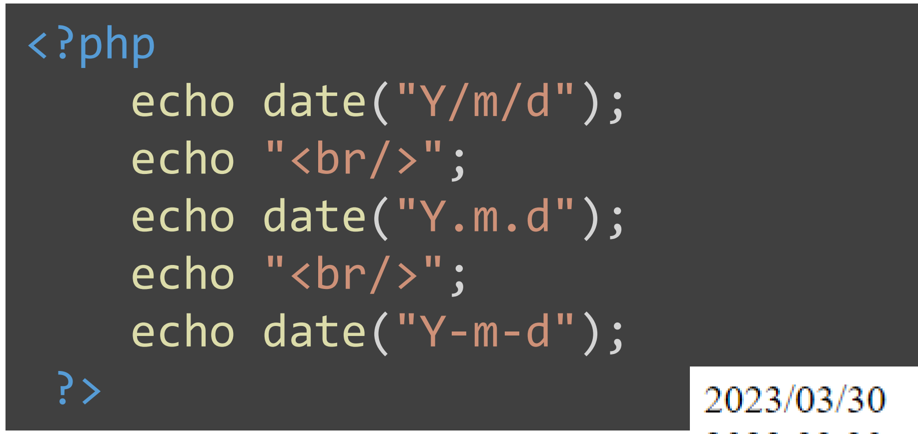


```
My name is Kai JimRefsnes.
My name is HegeRefsnes!
My name is StåleRefsnes...
```

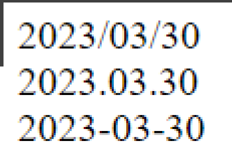
Functions – date()

- ➢ The first parameter in the date() function specifies how to format the date/time.
- ➢ It uses letters to represent date and time formats.
- ➢ Here are some of the letters that can be used:

- d -The day of the month (01-31)
- m -The current month, as a number (01-12)
- Y -The current year in four digits
- Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:



```
<?php
    echo date("Y/m/d");
    echo "<br/>";
    echo date("Y.m.d");
    echo "<br/>";
    echo date("Y-m-d");
 ?>
```



```
2023/03/30
2023.03.30
2023-03-30
```

Functions – timestamp

- ➢ The second parameter in the date() function specifies a timestamp.
- ➢ This parameter is optional.
- ➢ If you do not supply a timestamp, the current time will be used.
- ➢ In this example we use the mktime() function to create a timestamp for tomorrow.
- ➢ The mktime() function returns the Unix timestamp for a specified date.

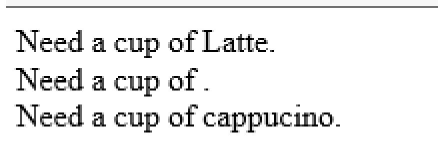
- Syntax
- mktime(hour,minute,second,month,day,year,is_dst)
- To go one day in the future we simply add one to the day argument of mktime()

```
<?php
    $tomorrow=mktime(0,0,0,date("m"),date("d")+1,date("Y"));
    echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

```
Tomorrow is 2023/03/31
```

Function - Example

```
<html>
<body>
<?php
function coffeetype($type = "Latte"){
    return "Need a cup of $type.";
}
echo coffeetype();
echo "<br/>";
echo coffeetype(null);
echo "<br/>";
echo coffeetype("cappucino");
?>
</body>
</html>
```



```
Need a cup of Latte.
Need a cup of .
Need a cup of cappucino.
```

Advantages of User Defined Functions

- Functions reduces the repetition of code within a program
- Functions makes the code much easier to maintain
- Functions makes it easier to eliminate the errors
- Functions can be reused in other application

Object Oriented Programming (OOP)

Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods.

Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

Objects in PHP

- A class represents an **Object,** with associated methods and variables
- To do something, we need to use the class
- Every object has its own local scope; $puppy1 and $puppy2 are entirely independent.
- The main power of classes is the inheritance; parent and child

$$\begin{tikzcd}
 & \text{Fruit} \arrow[dl] \arrow[dr] & \\
\text{Apple} & & \text{Orange}
\end{tikzcd}$$

- Child classes “inherit” all the methods and variables of the parent class
- Child classes can have their own methods and variables as well
- An object can also be destroyed using an **unset()** function

Object in PHP

```
 <?php

class dog {

public $name;

public function bark() {

echo 'Woof!' ;

 }

}

?>
```

```
class dog {
// defines the name of the class

public $name;
//define an object attribute (variable), the dog's name

public function bark() {
echo 'Woof!' ;
 }
//define an object action (function), the dog's bark

}
// end the class definition
```

Object in PHP

- Class:
- Defines the abstract characteristics of a thing (object), including the thing’s characteristics (its attributes, fields, or properties) and the thing’s behaviors (the things it can do, or methods, operations, or features). For example “**dog”** in the above example

- Object:
- Also refers as instance
- The Object is the actual object created at runtime. In the above example, **$name** is defined as the objects variable
- The object consists of state and the behavior that is defined in the object’s class
- Method:
- It is an object’s abilities
- Methods are similar to verbs
- In the above example “**bark**” is the method of the class dog

Object - $this

If you need to use the class variables within any class actions, use the special variable **$this. $this** refers to current object and can only be available inside the methods

```
<body>
    <?php
        class Fruit {
            // Properties
            public $name;
            public $color;
            // Methods
            function setName($name) {
                $this->name = $name;
            }
            function getName() {
                return $this->name;
            }
            function setColor($color) {
                $this->color = $color;
            }
            function getColor() {
                return $this->color;
            }
            }
            $apple = new Fruit();
            $apple->setName('Apple');
            $apple->setColor('Red');
            echo "Name: " . $apple->getName();
            echo "<br>";
            echo "Color: " .  $apple->getColor();
        ?>
 </body>
```

Object - Example

```php
<body>
  <?php
    class Person {
      public $fullname = false;
      public $givenname = false;
      public $familyname = false;
      public $room = false;
      function getName() {
        if ( $this->fullname !== false ) return $this->fullname;
        if ( $this->familyname !== false && $this->givenname !== false ) {
          return $this->givenname . ' ' . $this->familyname;
        }
        return false;
      }
    }
    $chuck = new Person();
    $chuck->fullname = "Adam Johnson";
    $chuck->room = "4429NQ";
    $colleen = new Person();
    $colleen->familyname = 'Holmes';
    $colleen->givenname = 'Ryan';
    $colleen->room = '3439NQ';
    print $chuck->getName() . "\n <br><br>";
    print $colleen->getName() . "\n";
  ?>
  </body>
</html>
```

```
Adam Johnson

Ryan Holmes
```

Constructor

- When creating a new object, you can pass a list of arguments to the class being called.
- These are passed to a special method within the class, called the *constructor*, which initializes various properties.
- If you create a __construct() function, PHP will automatically call this function when you create an object from a class.
- Construct function starts with two underscores (__)
- Constructor saves from calling *setName()*

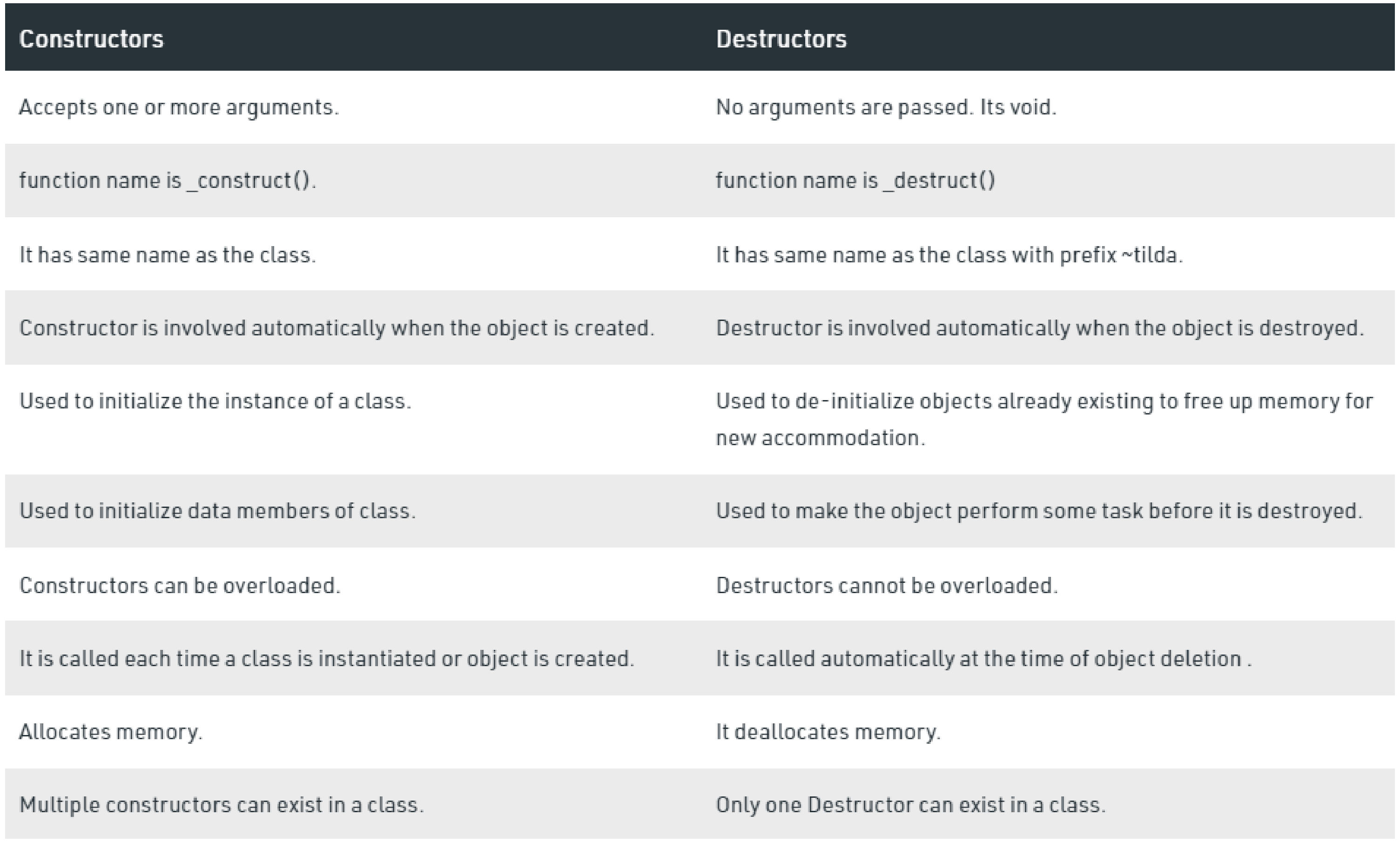
```
<html>
    <body>
        <?php
            class Fruit {
                public $name;
                public $color;
                function __construct($name) {
                    $this->name = $name;
                }
                function getName() {
                    return $this->name;
                }
                }

                $apple = new Fruit("Apple");
                echo $apple->getName();
        ?>
    </body>
</html>
```

Destructor

- The destructor can release a connection to database or other resources that is reserved within the class
- Destructor is created by using two underscores __*destruct*
- When a destructor function is created, PHP will call this function at the end of script
- Destructors give chance to objects to free up memory allocation , so that enough space is available for new objects or free up resources for other tasks.
- Destructor and Constructor reduces the amount of code

```
<!DOCTYPE html>
<html>
    <body>
        <?php
        class Fruit {
            public $name;
            public $color;
            function __construct($name) {
                $this->name = $name;
            }
            function __destruct() {
                echo "The fruit is {$this-
>name}.";
            }
        }
        $apple = new Fruit("Apple");
        ?>
    </body>
</html>
```

Difference between Constructor and Destructor

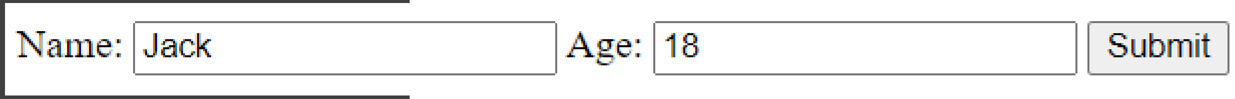


Constructors	Destructors
Accepts one or more arguments.	No arguments are passed. Its void.
function name is _construct().	function name is _destruct()
It has same name as the class.	It has same name as the class with prefix ~tilda.
Constructor is involved automatically when the object is created.	Destructor is involved automatically when the object is destroyed.
Used to initialize the instance of a class.	Used to de-initialize objects already existing to free up memory for new accommodation.
Used to initialize data members of class.	Used to make the object perform some task before it is destroyed.
Constructors can be overloaded.	Destructors cannot be overloaded.
It is called each time a class is instantiated or object is created.	It is called automatically at the time of object deletion .
Allocates memory.	It deallocates memory.
Multiple constructors can exist in a class.	Only one Destructor can exist in a class.

PHP Forms

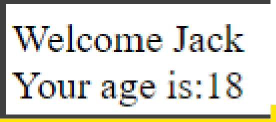
- The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.
- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

```
<html>
    <body>
        <form action="welcome.php" method="post">
            Name: <input type="text" name="name" />
            Age: <input type="text" name="age" />
            <input type="submit" />
        </form>
    </body>
</html>
```



Name: Jack Age: 18 Submit

```
<?php
    echo "Welcome ".$_POST["name"]."<br>";
    echo "Your age is:".$_POST["age"];
?>
```



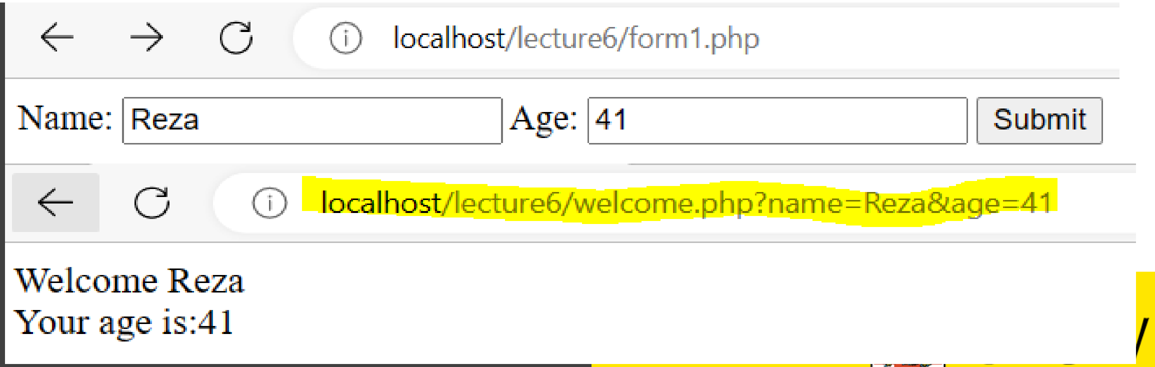
```
Welcome Jack
Your age is:18
```

PHP Forms

$_GET Variable

- The $_GET variable is an array of variable names and values sent by the HTTP GET method.
- The $_GET variable is used to collect values from a form with method="get".
- Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).
- When the user clicks the "Submit" button, the URL sent could look something like this: http://www.-int.com/welcome.php?name=Peter&age=3
- It is easier for a hacker to get information while using GET method
- GET method is not secure method and should not be used with passwords and sensitive information

```
<html>
    <body>
        <form action="welcome.php" method="get">
            Name: <input type="text" name="name" />
            Age: <input type="text" name="age" />
            <input type="submit" />
        </form>
    </body>
</html>
```


localhost/lecture6/form1.php

Name: Reza Age: 41 Submit

localhost/lecture6/welcome.php?name=Reza&age=41

Welcome Reza
Your age is:41

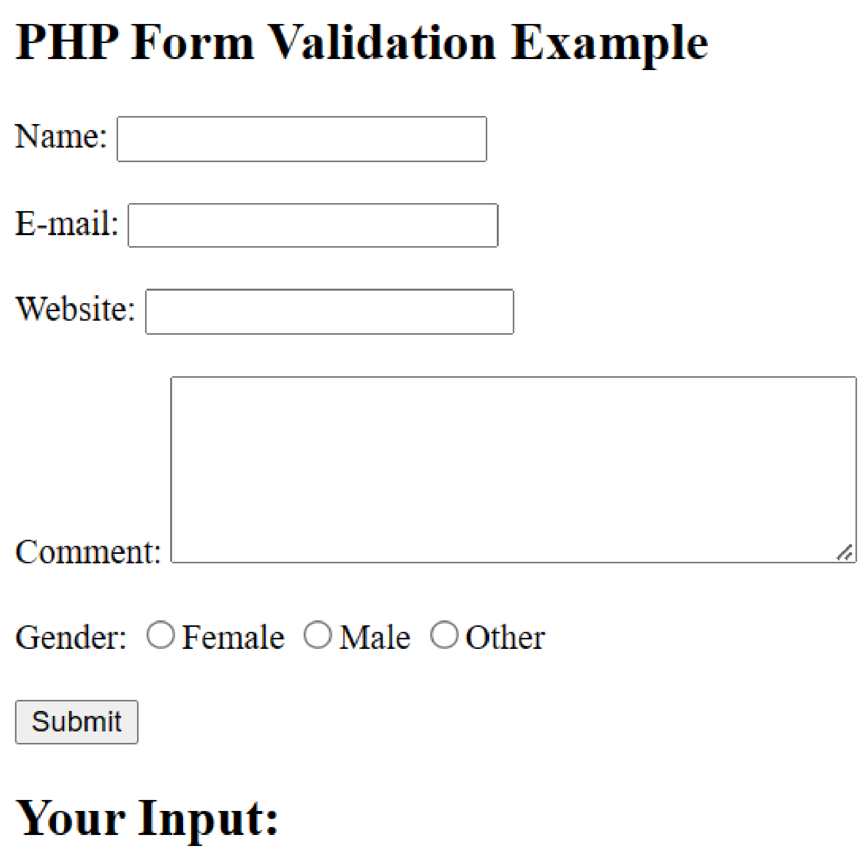
GET vs POST

- Both GET and POST create an array to hold key/value pairs
- Keys are the names of the form controls and values are the input data from the user
- $_GET is an array of variables passed to the current script via the URL parameters
  - Form data is visible to everyone
  - It is possible to bookmark the page
  - The limitation is about 2000 characters
  - May be used for sending non-sensitive data (no passwords)
- $_POST is an array of variables passed to the current script via the HTTP POST method
  - Information is invisible to everyone
  - No limits
  - Supports advanced functionality such as multi-part binary input while uploading files to server

Form Validation

Rules:

- Name: Required. + Must only contain letters and whitespace
- E-mail: Required. + Must contain a valid email address (with @ and .)
- Website: Optional. If present, it must contain a valid URL
- Comment: Optional. Multi-line input field (textarea)
- Gender: Required. Must select one



Form Validation - Form

Will be explained later

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") echo "checked";?>
     value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") echo "checked";?>
     value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) && $gender=="other") echo "checked";?>
     value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
```

Form Validation - Name

Regular expression

```
if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

Form Validation - Email

Built-in filter for validation emails

```
if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }
```

Form Validation - Website

Regular expression

```
if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also
   allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
   9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }
```

htmlspecialchars() function

The htmlspecialchars() function converts some predefined characters to HTML entities.

The predefined characters are:

- & (ampersand) becomes &amp;
- " (double quote) becomes &quot;
- ' (single quote) becomes &#039;
- < (less than) becomes &lt;
- > (greater than) becomes &gt;

This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

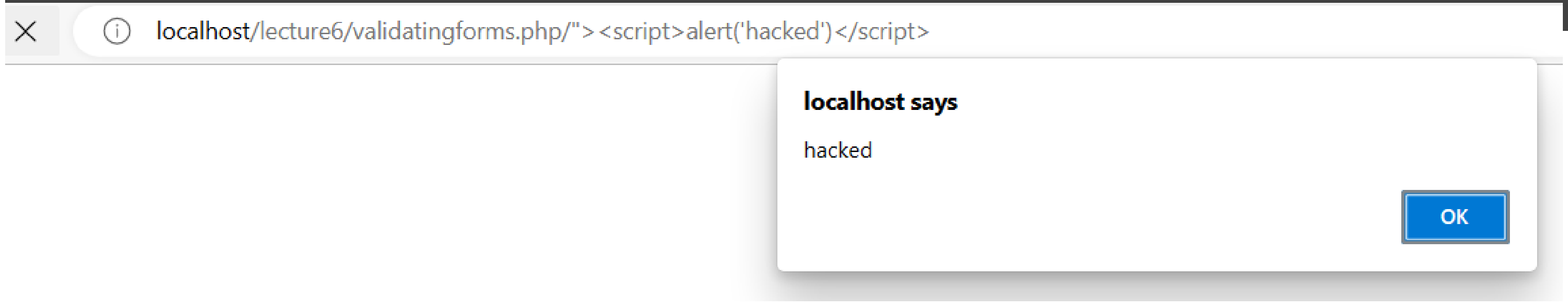
Form Security

If we don’t use htmlspecialchars

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Hackers can inject script codes through the URL:

```
http://localhost/lecture6/validatingforms.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

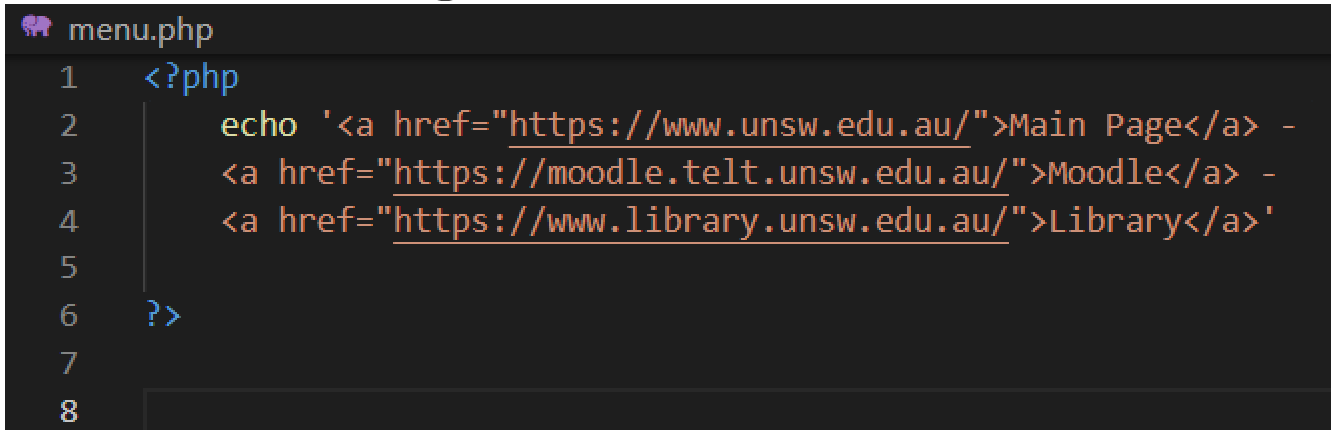


localhost/lecture6/validatingforms.php/"><script>alert('hacked')</script>

localhost says

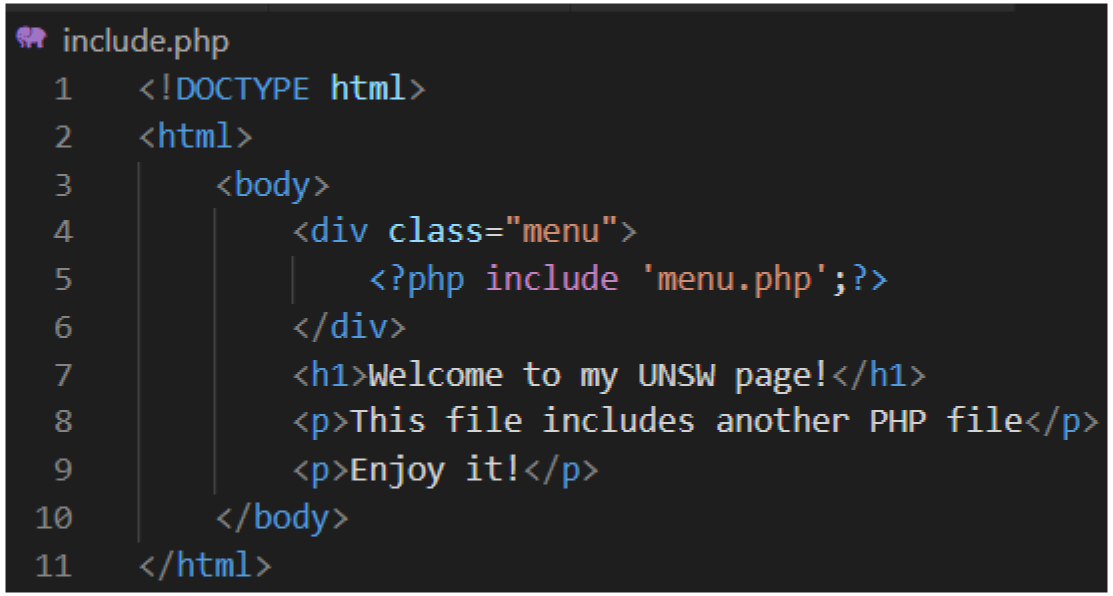
hacked

Including other files



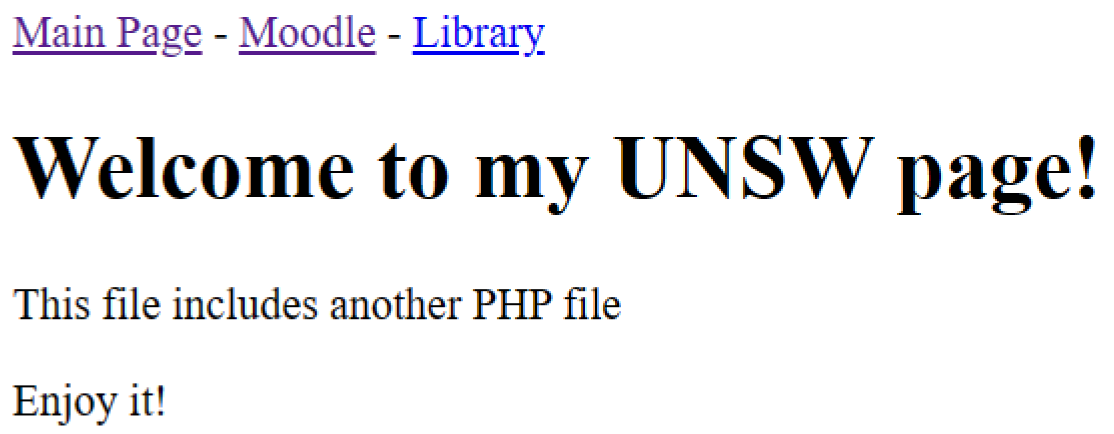
menu.php

```php
<?php
    echo '<a href="https://www.unsw.edu.au/">Main Page</a> -
    <a href="https://moodle.telt.unsw.edu.au/">Moodle</a> -
    <a href="https://www.library.unsw.edu.au/">Library</a>'

?>


```



include.php

```php
<!DOCTYPE html>
<html>
    <body>
        <div class="menu">
            <?php include 'menu.php';?>
        </div>
        <h1>Welcome to my UNSW page!</h1>
        <p>This file includes another PHP file</p>
        <p>Enjoy it!</p>
    </body>
</html>
```



Main Page - Moodle - Library

Welcome to my UNSW page!

This file includes another PHP file

Enjoy it!

File Handling

- Reading a file:

```
echo readfile("readme.txt");//or

$file = fopen("readme.txt", "r") or die("Unable to open file!");
echo fread($file,filesize("readme.txt"));
fclose($file);
```

- Create a file

- `$file = fopen("test.txt", "w"); // Use a for append`

- Writing to a file

```
$txt = “ZEIT3119\n";
fwrite($file, $txt);
$txt = “UNSW\n";
fwrite($file, $txt);
fclose($file);
```

- Deleting a file

```
unlink("test.txt");
```

Uploading Files

- Set the file_uploads directive on in "php.ini " (`file_uploads = On`)
- `Create uploads folder under the current folder of your file`

```html
<!DOCTYPE html>
<html>
    <body>
        <form action="upload.php" method="post" enctype="multipart/form-data">
        Select image to upload:
        <input type="file" name="fileToUpload" id="fileToUpload">
        <input type="submit" value="Upload Image" name="submit">
        </form>
    </body>
</html>
```

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =
strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check =
getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
  }
}

// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}
```

```php
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo "Sorry, only JPG, JPEG, PNG & GIF files are
allowed.";
  $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
  echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
  if
(move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$target_file)) {
    echo "The file ". htmlspecialchars( basename(
$_FILES["fileToUpload"]["name"])). " has been uploaded.";
  } else {
    echo "Sorry, there was an error uploading your file.";
  }
}
?>
```

UNSW CANBERRA

PHP Coockies

```
<?php
    $cookie_name = "user";
    $cookie_value = "Reza";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>
<html>
    <body>
        <?php
            if(!isset($_COOKIE[$cookie_name])) {
                echo "Cookie named '" . $cookie_name . "' is not set!";
            } else {
                echo "Cookie '" . $cookie_name . "' is set!<br>";
                echo "Value is: " . $_COOKIE[$cookie_name];
             }
        ?>
    </body>
</html>
```

PHP Session

```
<?php
    session_start();
?>
<!DOCTYPE html>
    <html>
    <body>
        <?php
            $_SESSION["favcolor"] = "green";
            $_SESSION["favanimal"] = "cat";
            echo "Session variables are set.";
        ?>
    </body>
</html>
```

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
    <body>
        <?php
            echo "Favorite color is " .
$_SESSION["favcolor"] . ".<br>";
            echo "Favorite animal is " .
$_SESSION["favanimal"] . ".";
        ?>
    </body>
</html>
```

This is not a browser session, i.e., the data is stored on the server about user's session. When the browser is closed the data is wiped off.

Final Note

- Project 1 deliverables D3 and D4 are due Friday, 7th April 2023 23h59.
- Two links for submissions:
  - **D3:** User Evaluation and Individual Reflection
  - **D4:** Final Project Code (zip file), Group Report (PDF or MS Word), and GitHub Repository
    - **Make sure your GitHub repository for Project 1 is either public or shared with me (rezarafeh)**

Individual Report Submission

Project 1 Group Submission