Forms
(Getting data from users)

Samar Abdelghani

Demonstrator, CS Department

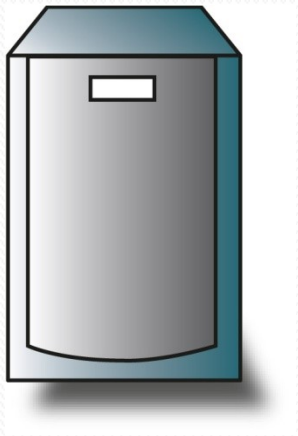
Forms: how they work

- We need to know..

1. How forms work.
2. How to write forms in XHTML.
3. How to access the data in PHP.

1.How forms work

User requests a particular URL



Web Server

XHTML Page supplied with Form

User fills in form and submits.
Another URL is requested and the
Form data is sent to this page either in
URL or as a separate piece of data.

User

XHTML Response

2 - How to write forms in XHTML.

```
<form action = ".."
      method = "...">

...
</form>
```

- action = URL
- method = get | post

http://www.w3schools.com/tags/tag_form.asp

XHTML Form

- The form is enclosed in form tags..

```
<form action="path/to/submit/page"
  method="get">
<!-- form contents -->
</form>
```

Form tags

- **`action="…"`** is the page that the form should submit its data to.
- **`method="…"`** is the method by which the form data is submitted. The option are either **`get`** or **`post`**. If the method is get the data is passed in the url string, if the method is post it is passed as a separate file.

Form fields: text input

- Use a text input within form tags for a single line freeform text input.

```
<label for="fn">First Name</label>
<input type="text"
       name="firstname"
       id="fn"
       size="20"/>
```

Form tags

- **`name="…"`** is the name of the field. You will use this name in PHP to access the data.
- **`id="…"`** is label reference string – this should be the same as that referenced in the **`<label>`** tag.
- **`size="…"`** is the length of the displayed text box (number of characters).

Form fields: password input

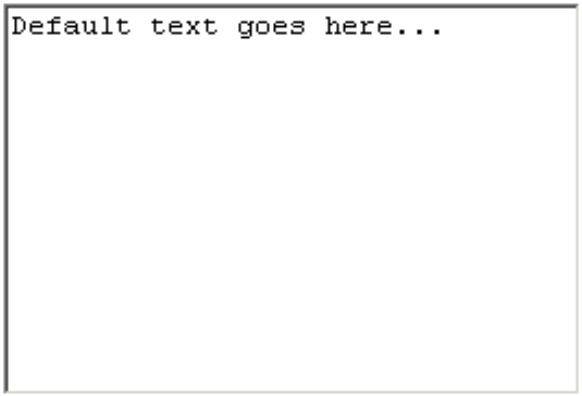
- Use a starred text input for passwords.

```
<label for="pw">Password</label>
<input type="password"
       name="passwd"
       id="pw"
       size="20"/>
```

Form fields: text input

- If you need more than 1 line to enter data, use a textarea.

```
<label for="desc">Description</label>
<textarea name="description"
          id="desc"
          rows="10" cols="30">
Default text goes here…
</textarea>
```



Form fields: text area

- **`name="…"`** is the name of the field. You will use this name in PHP to access the data.
- **`id="…"`** is label reference string – this should be the same as that referenced in the **`<label>`** tag.
- **`rows="…" cols=".."`** is the size of the displayed text box.

Form fields: drop down

```
<label for="tn">Where do you live?</label>
<select name="town" id="tn">
<option value="swindon">Swindon</option>
<option value="london”
        selected="selected">London</option>
<option value=“bristol">Bristol</option>
</select>
```

Form fields: drop down

- **`name="…"`** is the name of the field.
- **`id="…"`** is label reference string.
- **`<option value="…"`** is the actual data sent back to PHP if the option is selected.
- **`<option>…</option>`** is the value displayed to the user.
- **`selected="selected"`** this option is selected by default.

Form fields: radio buttons

```
<input type="radio"
      name="age"
      id="u30“
      checked=“checked”
      value="Under30" />
<label for="u30">Under 30</label>
<br />
<input type="radio"
      name="age"
      id="thirty40"
      value="30to40" />
<label for="thirty40">30 to 40</label>
```

Form fields: radio buttons

- **`name="…"`** is the name of the field. All radio boxes with the same name are grouped with only one selectable at a time.
- **`id="…"`** is label reference string.
- **`value="…"`** is the actual data sent back to PHP if the option is selected.
- **`checked="checked"`** this option is selected by default.

Form fields: check boxes

```
What colours do you like?<br />
<input type="checkbox"
       name="colour[]"
       id="r"
       checked="checked"
       value="red" />
<label for="r">Red</label>
<br />
<input type="checkbox"
       name="colour[]"
       id="b"
       value="blue" />
<label for="b">Blue</label>
```

Form fields: check boxes

- **`name="…"`** is the name of the field. Multiple checkboxes can be selected, so if the button are given the same name, they will overwrite previous values. The exception is if the name is given with square brackets – an array is returned to PHP.
- **`id="…"`** is label reference string.
- **`value="…"`** is the actual data sent back to PHP if the option is selected.
- **`checked="checked"`** this option is selected by default.

Submit button..

- A submit button for the form can be created with the code:

```
<input type="submit"
       name="submit"
       value="Submit" />
```

3.How to access the data in PHP.

In PHP…

- Data stored in a variable
- Depends on submission method:
  - GET
  - POST
- The variables are available in two superglobal arrays created by PHP called $_POST and $_GET.

GET and POST Method

- A form data can be submitted using POST and GET method
- Both are used for same purpose but stand apart for some specifications
- GET and POST create an array which holds key/value pairs, where keys are the name of the form controls and values are the input data by the user

GET and POST Method

- Both GET and POST method are treated as $_GET and $_POST in PHP
- These methods are superglobals, which means that they are always accessible, and they can be accessed using any function, class or file
- The $_GET method is an associative array of variables passed to the current script via the URL parameters

GET and POST Method

- The $_POST method is an array of variables passed to the current script via the HTTP POST method
- In this method the information is transferred in a hidden manner
- A form that submits information is appended to the URL in the form of *Query String* which consists of name = value pairs in URL known as *URL Encoding*

GET and POST Method

- This string contains user values/data, which are joined using equal (=) signs, separated by ampersand (&), and spaces are removed and replaced with plus (+) sign

Name1=value1&name2=value2&name3=value3

Get Method

- http://localhost/welcome.php?name=ahmed&email=aa@yahoo
- The code below uses **method="get"** for user to

The `action` attribute of the `form` element indicates that when the form data will be post

A unique `name` (e.g., `email`) is assigned to each of the form's `input` fields. When **SUBMIT** is clicked, each field's `name` and `value` are sent to the Web server.

```
7  <form action="welcome.php" method="get">
8  your name:<input type="text" name="name" />
9  your Email:<input type="text" name="email" />
10 <input type="submit" value="submit" name="submit" />
11 </form>
```

your name: your Email: Submit Query

Access data

- Access submitted data in the relevant array for the submission type, using the input name as a key.

```
<form action="path/to/submit/page"
      method="get">
<input type="text" name="email">
</form>

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

Get Method

- The code below is the server-side PHP script (welcome.php) where, $_GET associative array is used to receive sent information from server end

```php
<?php
echo "welcome:" . $_GET['name'] . "<br />";
echo "your email is:" . $_GET['email'] . "<br />";
?>
```

Output:

welcome:ahmed
your email is:aa@yahoo.com

Post Method

<form action="#" method="post">

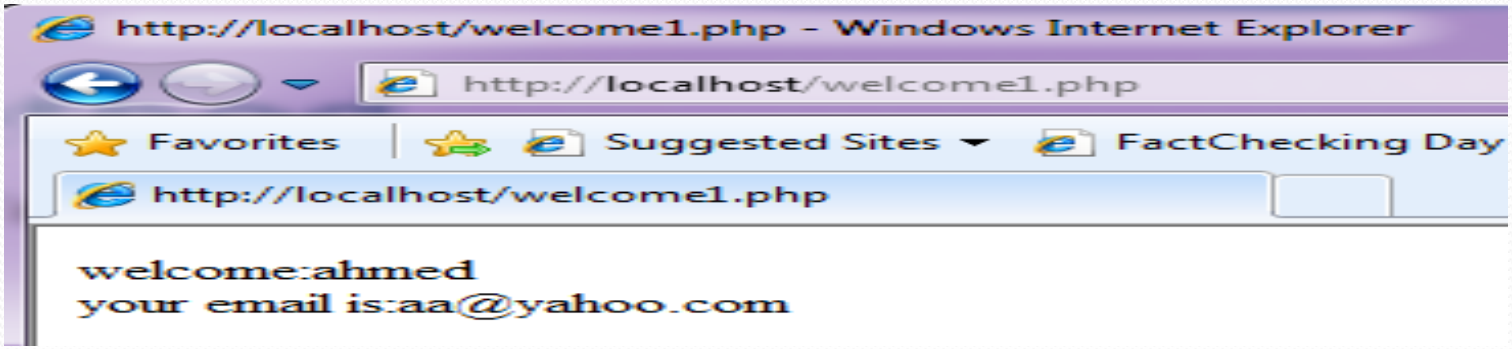
....

</form>

Below is a server-side PHP script where $_POST associative array is used to receive sent information at server end

```php
<?php
echo "welcome:" . $_POST['name'] . "<br />";
echo "your email is:" . $_POST['email'] . "<br />";
?>
```

Output:



Is it submitted?

- We also need to check before accessing data to see if the data is submitted, use isset() function.

```
if (isset($_POST['username'])) {
  // perform validation
}
```

Is it submitted?

Ex:

```
if (isset($_POST['myCheckbox2']))
  echo "Option 2 was selected";
else
  echo "Option 2 wasn't selected.";
```

2. Check which button was pressed

- Same as above

```
if (isset($_POST['button1']))
  echo "Button 1 was pressed";
elseif (isset($_POST['button2']))
  echo "Button 2 was pressed";
else
  echo "no button pressed";
```

A useful tip..

- I find that storing the validated data in a different array to the original useful.
- I often name this array ‘clean’ or something similarly intuitive.
- I then *only* work with the data in $clean, and never refer to $_POST/$_GET again.

Example

```
$clean = array();

if (ctype_alnum($_POST['username']))

{

$clean['username'] = $_POST['username'];

}
```

```php
$clean = array();
```

Initialise an array to store filtered data.

```
if (ctype_alnum($_POST['username']))
```

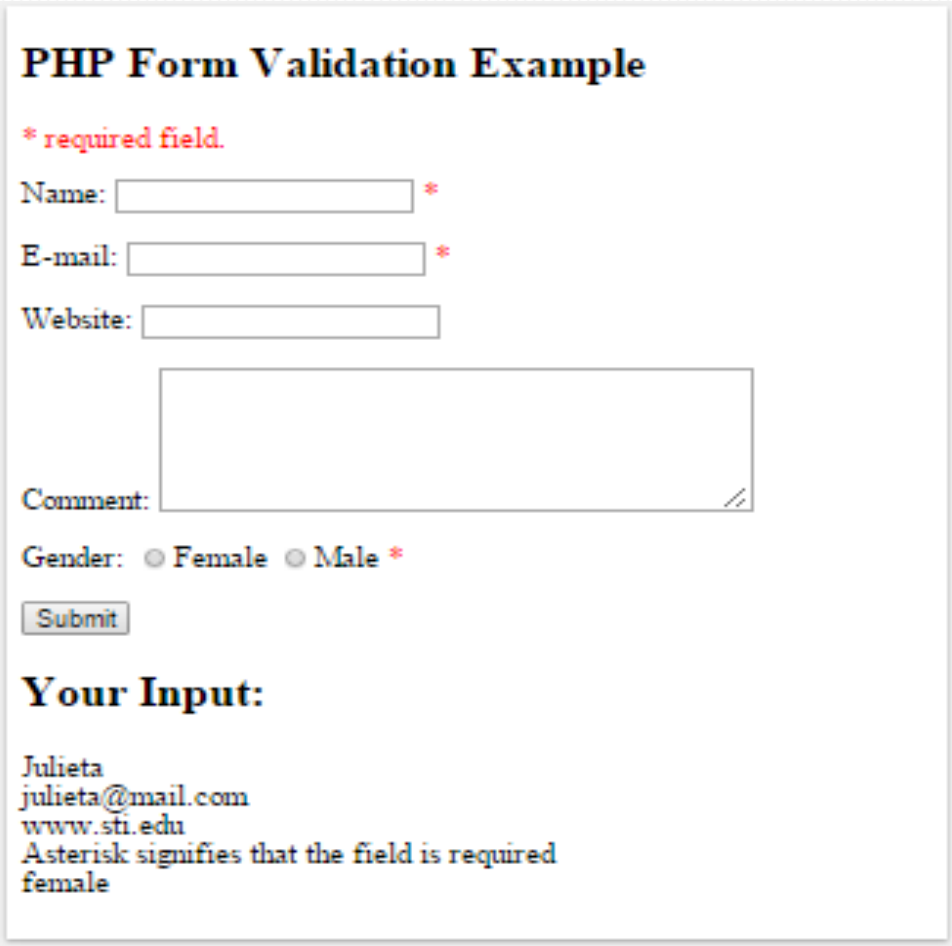
Inspect username to make sure that it is alphanumeric.

```
$clean['username'] = $_POST['username'];
```

If it is, store it in the array.

Lab Activity

Form validation



Form validation

The form consists of the following elements:

- Name (required field - must contain letters and whitespaces)
- E-mail (required field - must contain valid email address)
- Website (optional field - if present, must contain valid website URL)
- Comments (optional field - a multi-line text field)
- Gender (required field - must select a radio button)

Form Elements

- The *Name*, *E-mail*, *Website* are input elements
- *Input elements*, in particular, used *text* and *submit* values for its *types* attribute in order to create text fields and buttons
- The HTML code:

```
<?php
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
?>
```

Form Elements

- *Radio button* shows several options to the users from which the user may select one
- HTML Code:

```
<?php
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male

?>
```

Form Elements

- The *text area* is typically a large text field with multiple rows
- The *textarea* element has three attributes – *name*, *rows*, and *cols* attribute
- HTML code:

```
<?php
Comment: <textarea name="comment" rows="8" cols=
"80">Type your comments here...</textarea>
?>
```

Form Element

- The HTML code of the form element:

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

- when the form is submitted, the form data is sent with method=”post”
- So, the $_SERVER["PHP_SELF"] sends the submitted forms data to the page itself, instead of jumping to a different page

Form Element

- The *$_SERVER["PHP_SELF"]* is a super global variable that returns the filename of the currently executing script
- *Htmlspecialchars()* function converts special characters to HTML entities
- *Cross-site scripting (XSS)* is a type of computer security vulnerability typically found in Web application

Form Element

- Example: test_form.php

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

- if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<?php
<form method="post" action="test_form.php">
?>
```

Form Element

- consider that if a user enters the following URL in the address bar:

  http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E

- will be translated to:

```
<?php
<form method="post" action="test_form.php/"><script>
alert('hacked')</script>
?>
```

Form Element

be aware that any JavaScript code can be added inside the **<script>** tag

A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user's data

how to avoid $_SERVER[“PHP_SELF”] exploit?

- The $_SERVER[“PHP_SELF”] exploit can be avoided using the htmlspecialchars() function
- if the user tries to exploit the PHP_SELF variable, it will result:

```
<form method="post" action=
"test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&
lt;/script&gt;">
```

Validate Form Data with PHP

- The very first thing to do to validate form data with PHP is to pass all variables through PHP's htmlspecialchars() function
- For example:

```
<script>location.href('http://www.hacked.com')</script>
```

- With htmlspecialchars() function it would not be executed, because it would be saved as HTML escaped code like this:

```
&lt;script&gt;location.href('http://www.hacked.com')&lt
;/script&gt;
```

- **test_input()**

```php
<?php
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website =  test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Form Required Fields

- In the previous slide, all input fields were optional, meaning no required fields to be filled in by the user
- Here is a simple PHP script that checks the name for empty input and throws an error message if the input

```php
<?php
$name = $email = $gender = $comment = $website = "";
if (empty($_POST["name"])) {
    $nameErr = "Please enter your name";
  } else {
    $name = test_input($_POST["name"]);
  }
?>
```

Form Required Fields

To display the error message in the HTML form (this will be generated if the user tries to submit the form without filling in the required fields) use the code be

```
Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
```