Study Unit 11: PHP FORM CONTROLS AND PROCESSING

Outline

- Server Browser Interactions
- HTML Forms
- Submitting data
- Processing Form data in PHP

Learning Outcomes of Study Unit 11

You will learn the about PHP Registration Form with different parameters & form controls.

1.1: More HTML Forms

- reset, hidden
- fieldset, legend

1.2: Submitting Data

- Value (radio, option)
- URL-encoding
- GET vs. POST
- Uploading files

1.3: Processing Form data in PHP

- Superglobal arrays
- Process uploaded files

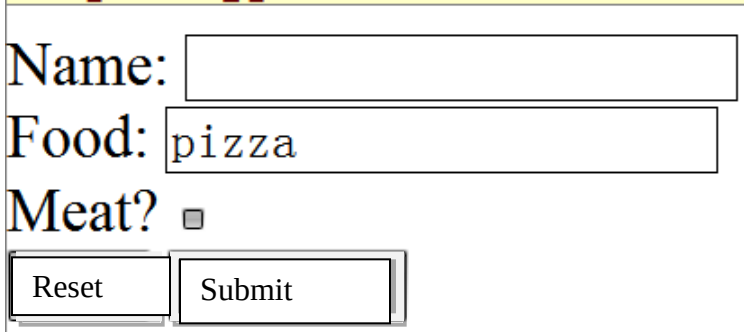
Outline

- **1.1: More HTML Forms**
- **Submitting Data**
- Processing Form data in PHP

Reset Buttons

```HTML
Name: <input type="text" name="name" /> <br />
Food: <input type="text" name="meal" value="pizza" /> <br />
<label>Meat? <input type="checkbox" name="meat" /></label> <br />
<input type="reset" />
```

output



- When clicked, returns all form controls to their initial values
- Specify custom text on the button by setting its value attribute

Grouping Input: <fieldset>, <legend>

groups of input fields with optional caption (block)

```HTML
<fieldset>
  <legend>Credit cards:</legend>
  <input type="radio" name="cc" value="visa" checked="checked" /> Visa
  <input type="radio" name="cc" value="mastercard" /> MasterCard
  <input type="radio" name="cc" value="amex" /> American Express
</fieldset>
```

output

Credit cards:

◉ Visa ○ MasterCard ○ American Express

Submit

- fieldset groups related input fields, adds a border; legend supplies a caption

Styling Form Controls

```css
element[attribute="value"] {
  property : value;
  property : value;
  ...
  property : value;
}
```

CSS

```css
input[type="text"] {
  background-color: yellow;
  font-weight: bold;
}
```

CSS

Borat

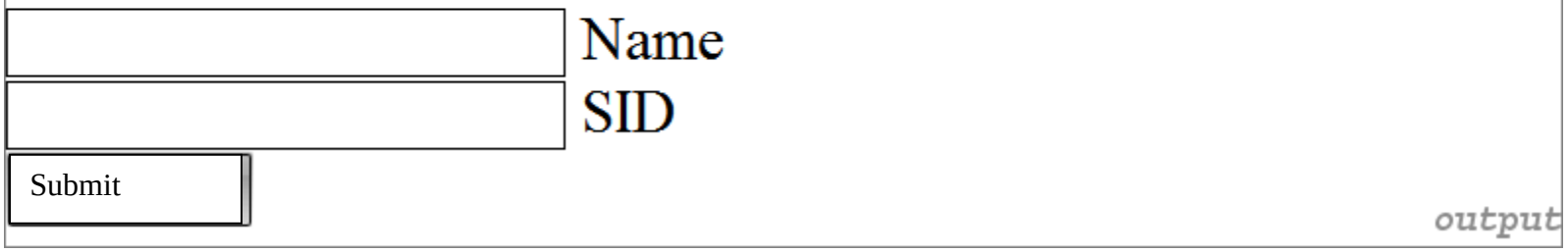
output

- **Attribute selector**: matches only elements that have a particular attribute value
- Useful for controls because many share the same element (input)

Hidden Input Parameters

```HTML
<input type="text" name="username" /> Name <br />
<input type="text" name="sid" /> SID <br />
<input type="hidden" name="school" value="UW" />
<input type="hidden" name="year" value="2048" />
```

output



- An invisible parameter that is still passed to the server when form is submitted
- Useful for passing on additional state that isn't modified by the user
- Especially useful when passing parameters between adjacent web requests

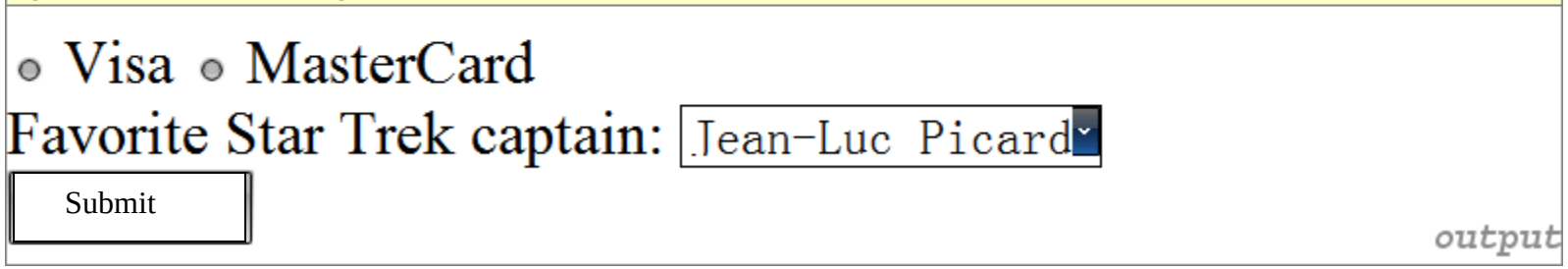
Outline

- More HTML Forms
- **1.2: Submitting Data**
- Processing Form data in PHP

Problems with Submitting Data

```
<label><input type="radio" name="cc" /> Visa</label>
<label><input type="radio" name="cc" /> MasterCard</label> <br />
Favorite Star Trek captain:
<select name="startrek">
  <option>James T. Kirk</option>
  <option>Jean-Luc Picard</option>
</select> <br />
```

HTML



output

- This form submits to our handy params.php tester page
- The form may look correct, but when you submit it...
- ***[cc] => on***, [startrek] => Jean-Luc Picard

The value Attribute

```
<label><input type="radio" name="cc" value="visa" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label> <br />
Favorite Star Trek captain:
<select name="startrek">
  <option value="kirk">James T. Kirk</option>
  <option value="picard">Jean-Luc Picard</option>
</select> <br />
```

HTML

output

- Value attribute sets what will be submitted if a control is selected
- [cc] => visa, [startrek] => kirk

URL-encoding

- Certain characters are not allowed in URL query parameters:
  - Examples: " ", "/", "=", "&"
- When passing a parameter, it is **URL-encoded** (reference table)
  - "Eric's cool!?" → "Eric*%27s*+cool*%3F%21*"
- You don't usually need to worry about this:
  - The browser automatically encodes parameters before sending them
  - The PHP $_REQUEST array automatically decodes them
  - ... but occasionally the encoded version does pop up (e.g. in Firebug)

Submitting Data to a Web Server

- Though browsers mostly retrieve data, sometimes you want to submit data to a server
  - Hotmail: Send a message
  - Flickr: Upload a photo
  - Google Calendar: Create an appointment
- The data is sent in HTTP requests to the server
  - With **HTML forms**
  - With **predefined URLs**
  - With **Ajax** (seen later)
- The data is placed into the request as parameters

HTTP GET vs. POST Requests

- **GET** : asks a server for a page or data if the request has parameters, they are sent in the URL as a query string

- **POST** : submits data to a web server and retrieves the server's response if the request has parameters, they are embedded in the request's HTTP packet, not the URL

  - For submitting data, a POST request is more appropriate than a GET **GET** requests **embed their parameters** in their URLs
  - URLs are **limited in length** (~ 1024 characters)
  - URLs cannot contain **special characters** without encoding
  - Private data in a URL can be seen

Form POST Example

```HTML
<form action="http://foo.com/app.php" method="post">
  <div>
    Name: <input type="text" name="name" /> <br />
    Food: <input type="text" name="meal" /> <br />
    <label>Meat? <input type="checkbox" name="meat" /></label> <br />
    <input type="submit" />
  <div>
</form>
```

output

Name:

Food:

Meat?

Submit

GET or POST?

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
  # process a GET request
  …
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
  # process a POST request
  …
}
```

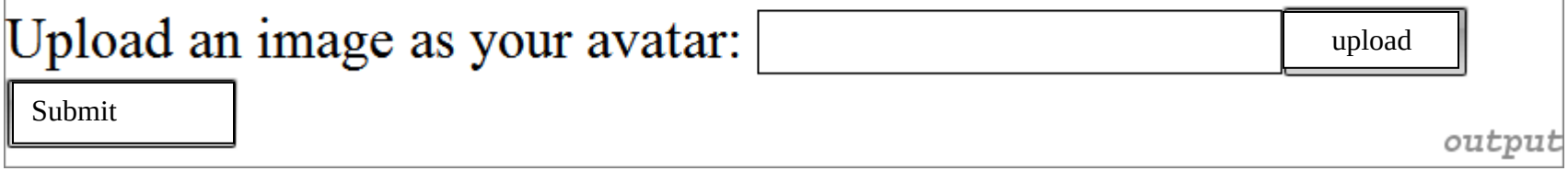
PHP

- Some PHP pages process both GET and POST requests
- To find out which kind of request we are currently processing, look at the —
  - Global **$_SERVER** array's "**REQUEST_METHOD**" element

Uploading Files

```
<form action="http://webster.cs.washington.edu/params.php"
      method="post" enctype="multipart/form-data">
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```

HTML



output

- Add a file upload to your form as an input tag with **type** of *file*
- Must also set the **enctype** attribute of the form
- What's going on exactly when you click the submit button?

Outline

- More HTML Forms
- Submitting Data
- **Processing Form data in PHP**

"Superglobal" Arrays

Array	Description
$_GET, $_POST	parameters passed to GET and POST requests
$_REQUEST	parameters passed to any type of request
$_SERVER, $_ENV	information about the web server
$_FILES	files uploaded with the web request
$_SESSION, $_COOKIE	"cookies" used to identify the user (seen later)

- PHP **superglobal** arrays (global variables) contain information about the current request, server, etc.
- These are special kinds of arrays called **associative arrays**.

Associative Arrays

```php
$blackbook = array();
$blackbook["marty"] = "206-685-2181";
$blackbook["stuart"] = "206-685-9138";
...
print "Marty's number is " . $blackbook["marty"] . ".\n";
```

- **Associative array** (a.k.a. **map**, **dictionary**, **hash table**) : uses **non-integer indexes** associates a particular index "key" with a value
  - i.e. key "marty" maps to value "206-685-2181"
- Syntax for embedding an associative array element in interpreted string:

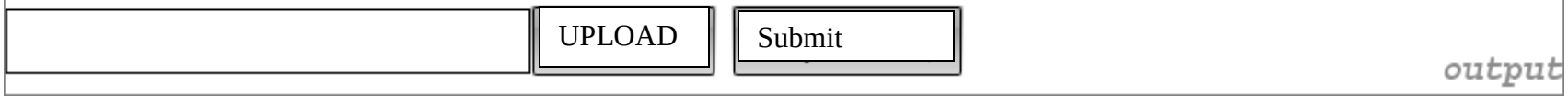
```php
print "Marty's number is {$blackbook['marty']}.\n";
```

Processing an Uploaded File in PHP

- Uploaded files are placed into global array **$_FILES**, **NOT** *$_REQUEST*
- Each element of **$_FILES** is itself an associative array, containing:
  - **name** : the local filename that the user uploaded
  - **type** : the MIME type of data that was uploaded, such as image/jpeg
  - **size** : file's size in bytes
  - **tmp_name** : a filename where PHP has temporarily saved the uploaded file
    - To permanently store the file, move it from this location into some other file

Uploading Details

```
<input type="file" name="avatar" />
```
HTML



output

- Example: if you upload borat.jpg as a parameter named avatar,
  - $_FILES["avatar"]["name"] will be "borat.jpg"
  - $_FILES["avatar"]["type"] will be "image/jpeg"
  - $_FILES["avatar"]["tmp_name"] will be something like "/var/tmp/phpZtR4TI"

Processing Uploaded File, Example

```php
$username = $_REQUEST["username"];
if (is_uploaded_file($_FILES["avatar"]["tmp_name"])) {
  move_uploaded_file($_FILES["avatar"]["tmp_name"], "$username/avatar.jpg");
  print "Saved uploaded file as $username/avatar.jpg\n";
} else {
  print "Error: required file not uploaded";
}
```

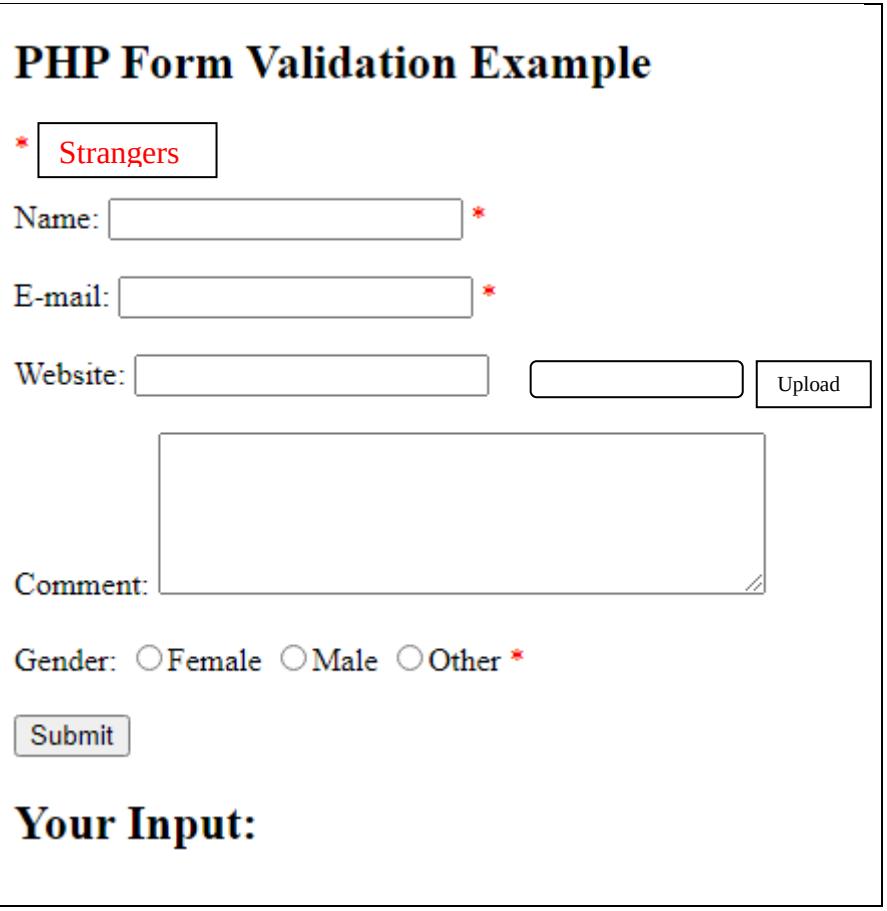
- Functions for dealing with uploaded files:
  - **is_uploaded_file**(*filename*)
    returns TRUE if the given filename was uploaded by the user
  - **move_uploaded_file**(*from*, *to*)
    moves from a temporary file location to a more permanent file
- Proper idiom: check is_uploaded_file, then do move_uploaded_file

Questions For Study Session 11

Now that you have completed this study unit, you can assess how well you have achieved its Learning Outcomes by answering these questions. Write your answers in your Study Diary and discuss them with your Tutor at the next Study Support Meeting or Online interactive sessions. You can also check your answers at the Self-Review Answers section which is located at the end of this Module.

1: Write a PHP Registration page.
Convert it to a PHP page, which
only shows data you submitted
at the bottom of this page when your first name is "Eric",
otherwise it shows the message "Strangers Forbidden!" instead

2: upload an image with registration form.



Self-Review Answers (SRA) for Study Unit 11

1: Write a PHP Registration page.Convert it to a PHP page, which only shows data you submitted at the bottom of this page when your first name is “Eric”, otherwise it shows the message “Strangers Forbidden!” instead.

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if ($_POST["name"]=="Eric") {
    $nameErr = "Strangers";
  } 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";
```

```
    }
  }

  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";
    }
  }

  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";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }
```

```
  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

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

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<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>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

2: upload an image with registration form.
Use this link for practice -> https://www.w3schools.com/php/php_file_upload.asp

References and Additional Reading Materials

- PHP home page: http://www.php.net/
- W3Schools PHP tutorial: http://www.w3schools.com/PHP/