Working with Forms

HTTP GET / POST, Validation, Escaping, Input Types, Submitting Arrays, URL Redirecting

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

UNIVERSITY of GREENWICH
Alliance with FPT Education

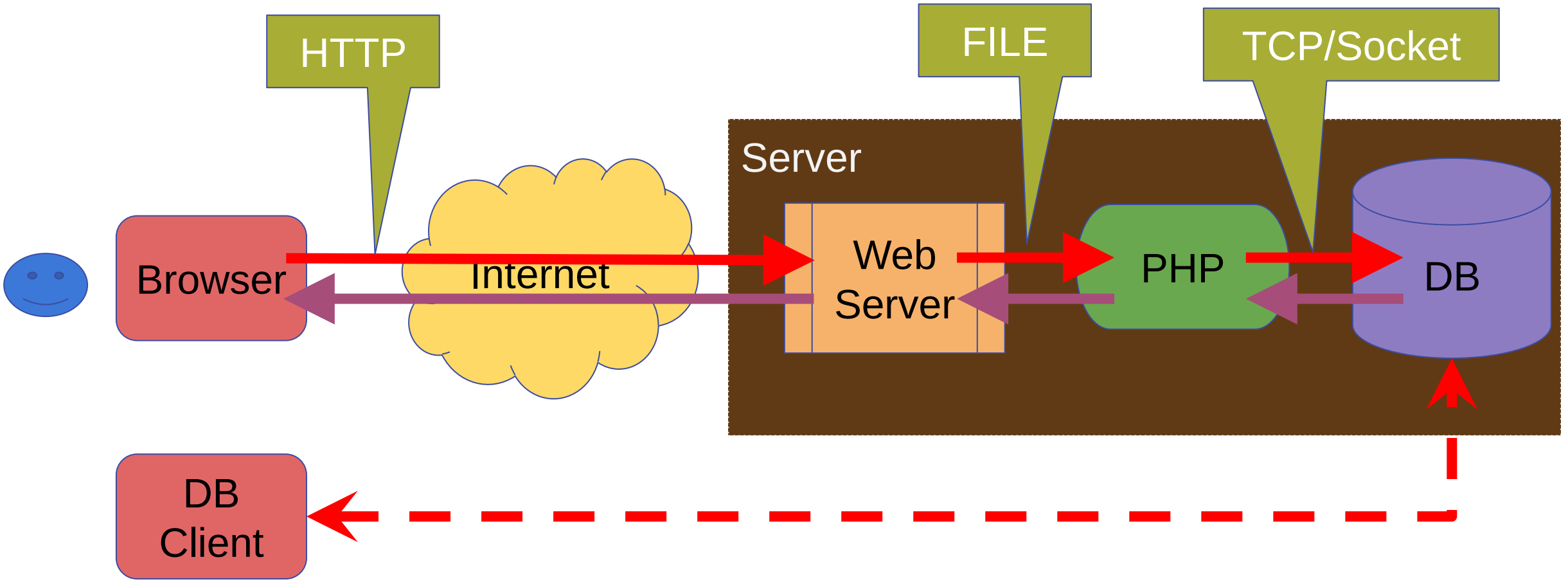
Table of Contents

HTTP

How Browsers talk with Servers?

UNIVERSITY of GREENWICH
Alliance with FPT Education

Big Picture



UNIVERSITY of GREENWICH
Alliance with FPT Education

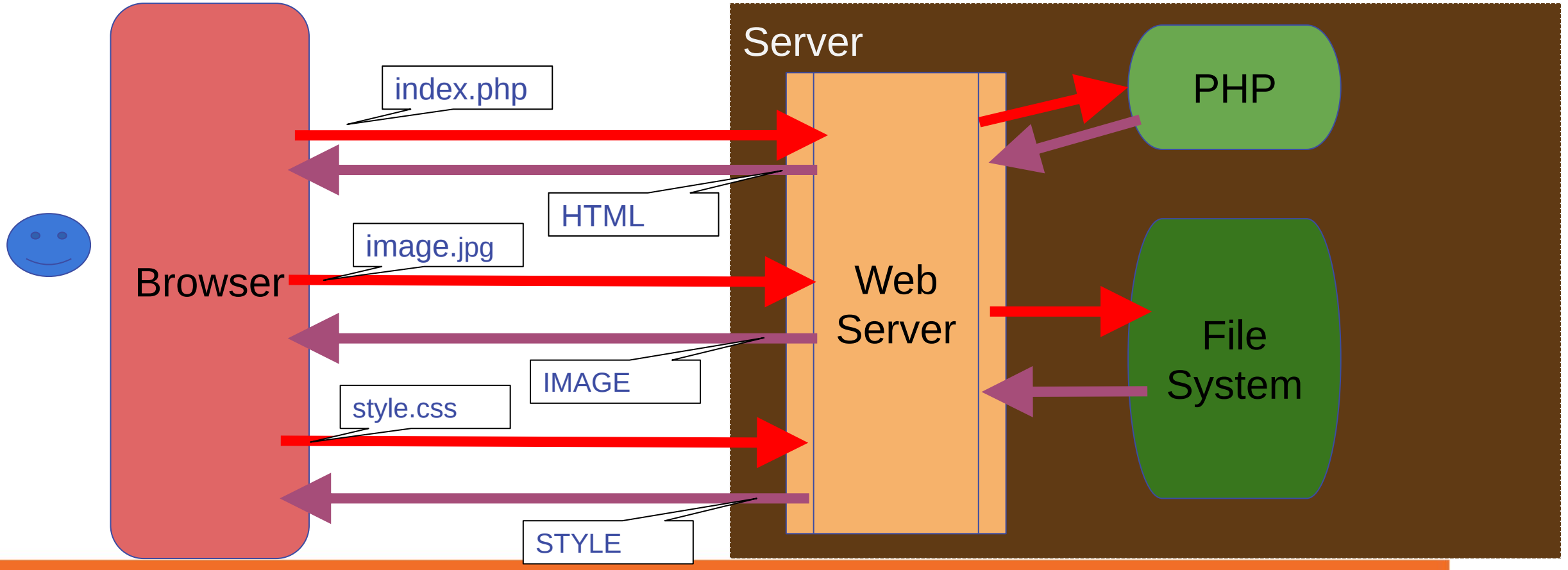
What is HTTP

HyperText Transfer Protocol

- Main protocol for WEB communication
- Base on client-server principe
- Text based - easy to read, easy to debug
- Use request/response principe

UNIVERSITY of GREENWICH
Alliance with FPT Education

Web Page Loading



HTTP REQUEST METHODS

How Browsers Send Form Data?

UNIVERSITY of GREENWICH
Alliance with FPT Education

HTTP Request Methods

- Forms allow the user to enter data that is sent to a server for processing via HTTP request methods
  - The most used HTTP request methods: **GET** and **POST**
- In PHP the posted form data is stored in the **$_GET** or **$_POST** associative arrays

GET Request Method

- HTTP GET
  - Retrieves data from the server from given URL
- The form data is stored in **`$_GET`** associative array
- The whole query string can be accessed using **`$_SERVER['QUERY_STRING']`** environment variable

```
<form method="get" action="index.php">
      …
</form>
```

GET Request Method – Example

```
<form method="get">
    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
```

```
<?php
// Check the keys "name" or "age" exist
if (isset($_GET["name"]) || isset($_GET["age"])) {
    echo "Welcome " . htmlspecialchars($_GET['name']) . ". <br />";
    echo "You are " . htmlspecialchars($_GET['age']). " years old.";
}
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

POST Request Method

- The **POST** method transfers data in the HTTP body
  - Not appended to the query string
- The posted data is stored in **$_POST** associative array
- By using **https://** you can protect your posted data
- POST can send text and binary data, e.g. upload files

```
<form method="post" action="index.php">
      …
</form>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

POST Request Method – Example

```
<form method="post">
    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
```

```
<?php
// Check the keys "name" or "age" exist
if (isset($_POST["name"]) || isset($_POST["age"])) {
    echo "Welcome " . htmlspecialchars($_POST['name']) . ". <br />";
    echo "You are " . htmlspecialchars($_POST['age']). " years old.";
}
?>
```

HTML ESCAPING & DATA VALIDATION

HTML Escaping: Motivation

- Suppose we run this PHP script:

```
<form method="get">
    Enter your name: <input type="text" name="name" />
    <input type="submit" />
</form>

<?php
if (isset($_GET["name"]))
   echo "Hello, " . $_GET["name"];
?>
```

- What if we enter the following in the input field?

```
<script>alert('hi')</script>
```

HTML Escaping: htmlspecialchars()

- **htmlspecialchars(string)**
  - Converts HTML special characters to entities: **&** **"** **'** **<** and **>** become **&** **&quote;** **'** **<** and **>**

```
<form method="get">
    Enter your name: <input type="text" name="name" />
    <input type="submit" />
</form>

<?php
if (isset($_GET["name"]))
   echo "Hello, " . htmlspecialchars($_GET["name"]);
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Principles of HTML Escaping

- How and when the HTML escape?
  - HTML escaping should be performed on all data printed in an HTML page, that could contain HTML special chars
  - Any other behavior is incorrect!
- Never escape data when you read it!
  - Escape the data when you print it in a HTML page
- Never store HTML-escaped data in the database!
- Never perform double HTML escaping

UNIVERSITY of GREENWICH
Alliance with FPT Education

Example of Correct HTML Escaping

- Sample form that can submit HTML special characters:

```
<form method="get">
  Name: <input type="text" name="name" value="&lt;br&gt;" />
  <input type="submit" />
</form>
```

- Example of **correct** HTML escaping (data only!):

```
<?php
if (isset($_GET["name"]))
   echo "Hi, <i>" . htmlspecialchars($_GET["name"] . "</i>");
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Example of Incorrect HTML Escaping

- Sample form that can submit HTML special characters:

```
<form method="get">
  Name: <input type="text" name="name" value="&lt;br&gt;" />
  <input type="submit" />
</form>
```

- Example of incorrect HTML escaping (don't escape everything):

```
<?php
if (isset($_GET["name"]))
   echo htmlspecialchars("Hi, <i>" . $_GET["name"] . "</i>");
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Data Normalization

- **addcslashes()** – escapes given list of characters in a string

```
echo addcslashes("say('hi')", ';|<>\'"');
// Result: say(\'hi\')
```

- **quotemeta()** – escapes the symbols **. \ + * ? [^] ($)**
- **htmlspecialchars()** – convert special characters to HTML entities, as seen in examples
- **htmlentities()** – escapes all HTML entities (**£** → **£**)
  - Escapes special symbols in a string: **&**, **"**, **'**, **<**, **>**

```
echo htmlentities("A '£' is <b>bold</b>");
// Outputs: A '&pound;' is &lt;b&gt;bold&lt;/b&gt;
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

PHP Automatic Escaping Engine - WARNING!

- PHP supported the **`magic_quotes`** engine
  - It escapes all necessary characters in the **`$_GET`**, **`$_POST`** and **`$_COOKIE`** array automatically
  - In versions before 5.2 it is turned on by default
  - Considered dangerous approach - deprecated in PHP 5.4
  - **DO NOT USE IT!!!**
  - Developers should handle escaping manually

UNIVERSITY of GREENWICH
Alliance with FPT Education

Validating User Input

- Data validation ensures the data we collect is correct
  - May be performed by **`filter_var()`** in PHP

```
<?php
$ip_a = '127.0.0.1';
$ip_b = '42.42';
if (filter_var($ip_a, FILTER_VALIDATE_IP)) {
    echo "This (ip_a) IP address is considered valid.";
}
if (filter_var($ip_b , FILTER_VALIDATE_IP)) {
    echo "This (ip_b) IP address is considered valid.";
}
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Validating User Input (2)

```
<form>
    <input type="text" name="num" />
    <input type="submit" />
</form>
<?php
if (isset($_GET['num'])) {
    $num = intval($_GET['num']);
    if ($num < 1 || $num > 100) {
        echo "Please enter an integer number in range [1..100].";
        die;
    }
    echo "You entered valid number: $num.";
}
?>
```

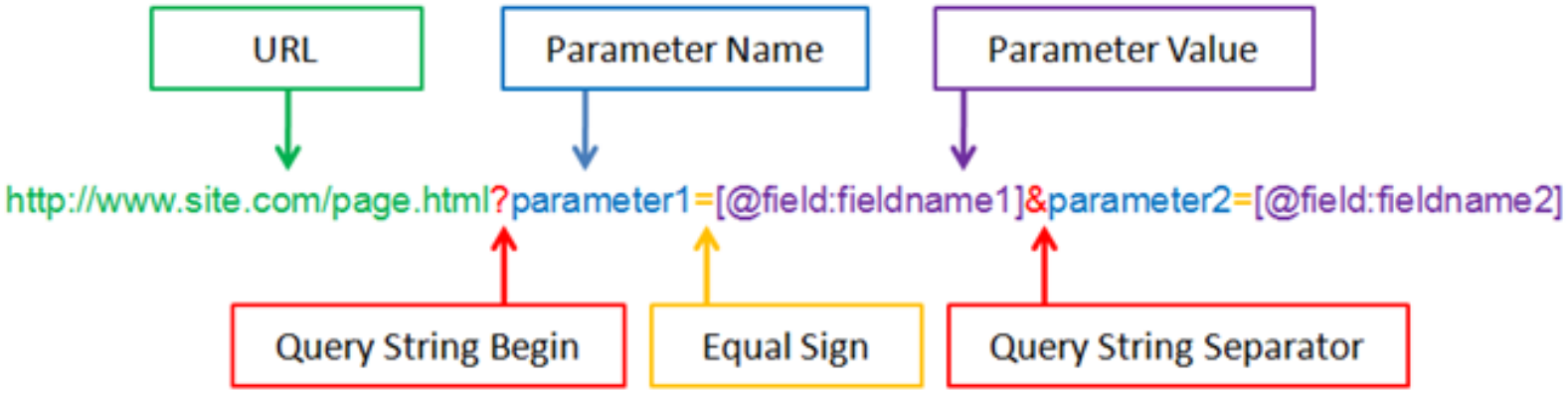
HTML ESCAPING & DATA VALIDATION

Live Demo

UNIVERSITY of GREENWICH

Alliance with FPT Education

QUERY STRING



UNIVERSITY of GREENWICH
Alliance with FPT Education

What is a Query String?

- A query string is a part of a **URL** following a question mark (**?**)
- Commonly used in searches and dynamic pages
- Accessed by **$_SERVER['QUERY_STRING']**

```
<form>
    <input type="text" name="firstName" />
    <input type="submit" />
</form>
<?php
echo $_SERVER['QUERY_STRING'];
?>
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Creating a Query String

- Most common way is by using a form with a **`GET`** method
- You can also use scripts to add to the query string or simply write your links with the query strings in the **`href`** attribute

WORKING WITH CHECKBOXES

UNIVERSITY of GREENWICH
Alliance with FPT Education

Checkoxes

- Checkboxes are created by setting an input with type "checkbox"

```
<input type="checkbox" name="two-way-ticket" />
```

- A checkbox is only submitted if it's actually checked

```
if (isset($_GET['two-way-ticket']) ) {
   echo "Two-way ticket";
} else {
   echo "One-way ticket";
}
```

UNIVERSITY of GREENWICH

Alliance with FPT Education

HIDDEN FIELDS

UNIVERSITY of GREENWICH
Alliance with FPT Education

Hidden Fields

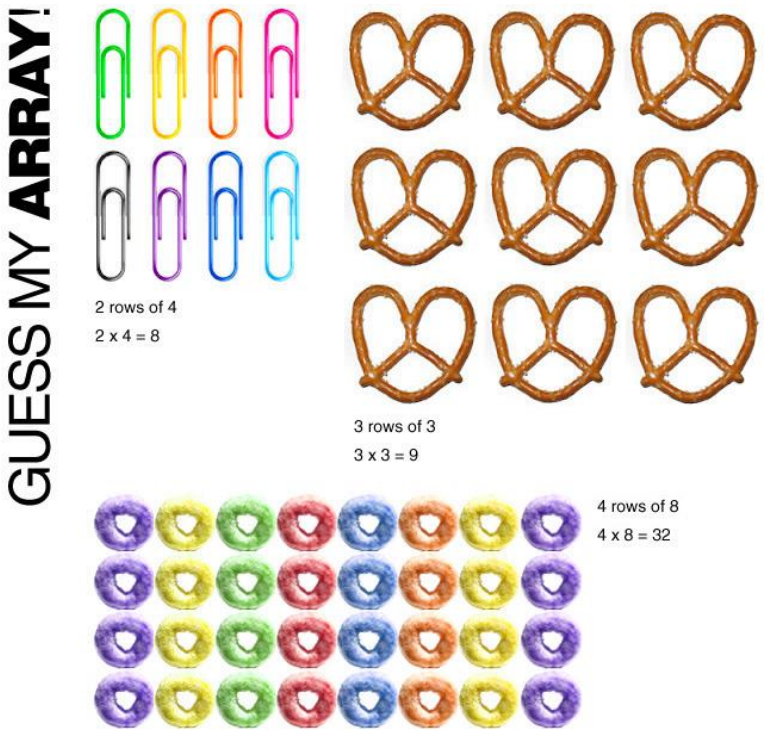
- Created by setting the type of input to hidden
- Submit information that is not entered by the user
- Not visible to the user, but visible with [F12]

```
<form method="post">
    <input type="text" name="user" />
    <input type="submit" />
    <?php if (isset($_POST['user'])) { ?>
    <input type="hidden" name="hiddenName"
        value="<?php echo sha1($_POST['user']) ?>" />
    <?php } ?>
</form>
```

Applies sha1 hashing to a string

UNIVERSITY of GREENWICH

Alliance with FPT Education



SUBMITTING ARRAYS

UNIVERSITY of GREENWICH
Alliance with FPT Education

Submitting Arrays

- In order for an input to be treated as an array, you must put brackets "**[ ]**" in the **name** attribute:

```
<form method="post">
    <select name="people[]" multiple="multiple">
        <option value="Mario">Mario</option>
        <option value="Svetlin">Svetlin</option>
        <option value="Teodor">Teodor</option>
    </select>
    <input type="submit" value="submit"/>
</form>
```

Submitting Arrays (2)

- The selected form elements come as an array:

```
<?php
if (isset($_POST['people'])) {
    foreach($_POST['people'] as $person) {
        echo htmlspecialchars($person) . '</br>';
    }
}
?>
```

UNIVERSITY of GREENWICH

Alliance with FPT Education

OTHER INPUT TYPES

UNIVERSITY of GREENWICH
Alliance with FPT Education

Other Input Types

- Radio, date, datetime, time, number, range, color, …

```
<form method="post">
    Male <input type="radio" name="gender" value="male" /> <br/>
    Female <input type="radio" name="gender" value="female" /> <br/>
    <input type="submit" value="submit"/>
</form>
<?php
if (isset($_POST['gender'])) {
    $selected_radio = $_POST['gender'];
    echo "Selected: $selected_radio";
}
?>
```

REDIRECTING THE BROWSER

UNIVERSITY of GREENWICH
Alliance with FPT Education

Redirecting the Browser

- Done by using the HTTP "**Location**" header

```
header('Location: http://greenwich.edu.vn');
```

- This sends HTTP 302 "Found" in the HTTP response status code
  - Tells the browser to open a new URL

UNIVERSITY of GREENWICH
Alliance with FPT Education

Summary

- HTTP request methods – **GET**, **POST**, etc.
- Normalization and validation
- Working with query strings
- You can easily combine **PHP** and **HTML**
- You can get input as array
- Special input fields – checkboxes, hidden fields