HTML and PHP Basics

Instructor
Davide Maiorca

Web Security and Malware Analysis

M.Sc. in Computer Engineering, Cybersecurity and Artificial Intelligence

University of Cagliari, Italy

Goals

- Understanding the basics of HTML and PHP codes
- Understanding how PHP integrates with HTTP requests and cookies
- Being able to run simple PHP codes

HTML

- Acronym for HyperText Markup Language
- The most used language to represent web pages
- Describes the structure of the web page and tells the browser how to render it
- HTML elements are repsented by *tags*
- Each HTML *tag* describes the element that is represented (e.g., body, paragraph, image…)
- It is easy to test HTML by copying and pasting it

First HTML Example

- *!DOCTYPE html* defines the type of document used (HTML)
- <*html*> is the *root* of the page
- <*head*> contains the basic meta-information about the page
  - For example, the title
- <*body*> represents the visualized contents of the HTML page
  - <*h1*> represents the page heading
  - <*p*> represents a paragraph
- Properties of the tag (e.g., colors) are defined by the *style* property
  - `<body style="background-color:powderblue;">`

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Links, Images, Buttons, Lists

- Links are defined by the <*a*> tag

```
<a href="https://www.w3schools.com">This is a
link</a>
```

- Images are defined by the <*img*> tag
  - *src* defines the source of the image
  - *alt* defines the text to show when the image is not found
  - *width* and *height* represent the size of the image

```
<img src="w3schools.jpg" alt="W3Schools.com" wi
dth="104" height="142">
```

- Buttons are defined by the tag <*button*>

```
<button>Click me</button>
```

- <*ul*> defines a list of unordered elements
  - <*li*> is the tag of each object
  - Ordered lists are defined by <*ol*>

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Blocks, Forms, Scripts

- Blocks represent an area of the file that starts with a newline and take the whole page width
  - Introduced by the <*div*> tag

- Forms are defined by the tag <*form*>
  - They allow users to send inputs to the web application
  - The requested input is defined by the tag <*input*> and the user can specify the type and the name
  - The tag <*value*> specifies the value that each field of the form takes
  - The input type *submit* allows to send data through the form to a webpage defined by the tag <*action*>
  - <*br*> defines a newline

- Scripts are defined by the tag <*script*>

```
<div style="background-
color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of
England.</p>
</div>

<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value=
"Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="
Mouse"><br><br>
  <input type="submit" value="Submit">
</form>

<script>
document.getElementById("demo").innerHTML =
"Hello JavaScript!";
</script>
```

PHP

- Acronym for PHP: Hypertext Preprocessor
- Widely-used, open source scripting language
- PHP scripts are executed on *web servers*
  - The result is returned as plain HTML
- PHP has multiple functionalities
  - It can generated dynamic content
  - It can create, open, read, write, delete, and close files on the server
  - It can collect form data
  - It can send and receive cookies
  - It can add, delete, modify data in databases
  - It can be used to control user-access
  - It can encrypt data

PHP – First Example

- PHP code is defined by the <?php … ?> tag
  - Files are saved with the .php tag
  - PHP code is contained in HTML code, which is also saved in .php files
  - Every PHP statement ends with ;
  - Comments are either expressed with // or /*
- PHP keywords (e.g., if, for, case) are NOT case sensitive
- Variables start with $
  - They must start with either underscore or a letter
  - They cannot start with numbers
  - You can include comments even between operations
  - PHP is loosely typed (no need to specify the variable type)
  - Strings are expressed with double quotes (and chained with *dots*
- You can use *echo* or *print* to print something

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
// My php code
echo "Hello World!";
$test 24 + /* test */ + 3;
echo $test;
?>

</body>
</html>
```

PHP – Conditions and Loops

- Conditions in PHP are expressed by the if, elseif, else statements
  - The program in the example uses the function *date* (which returns a string) to get the current hour (parameter *H)*
  - Remember the {…} brackets after the condition
  - *Switch-case* statements are also possible

- Loops can be of four types
  - *For*
  - *While*
  - *Do…while*
  - *Foreach*

- Standard *for* is essentially like the for loop in C

- *Foreach* is like the Python loop
  - In the example, it takes each value from the list

```php
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

```php
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

```php
<?php
$colors = array("red", "green");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

PHP – Functions and Arrays

- Functions are defined by the *function* keyword and they work as in Python
  - No type-enforcement is required
  - Default values like in Python can be used

```php
<?php
function familyName($fname) {
    echo "$fname <br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

- Arrays can be defined with the *array* keyword
  - Similar to Python arrays
  - Indexable

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

- Arrays in PHP can also be *associative*
  - Like dictionaries in Python
  - They can be accessed through names

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

PHP – Handling Forms

- PHP is typically used to parse HTTP requests sent through forms
- There are two special variables that handle requests
  - $_GET for GET requests
  - $_POST for POST requests
  - These are *superglobal* variables (accessed from everywhere)
  - $_GET and $_POST are associative arrays
- In this example, the client employs a HTTP GET request to embed parameters put with a form
  - The name of the parameters are defined by the *name* tag
  - Parameters in this request are embedded in the URLs
- Typically, parameters are sent through POST requests
  - Not visible to users from the URL

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
<html>
<body>

Welcome <?php echo $_GET["name"];
?><br>
Your email address is: <?php echo
$_GET["email"]; ?>

</body>
</html>
```

PHP – Handling Forms

- PHP is typically used to parse HTTP requests sent through forms
- There are two special variables that handle requests
  - $_GET for GET requests
  - $_POST for POST requests
  - These are *superglobal* variables (accessed from everywhere)
  - $_GET and $_POST are associative arrays
- In this example, the client employs a HTTP GET request to embed parameters put with a form
  - The name of the parameters are defined by the *name* tag
  - Parameters in this request are embedded in the URLs
- Typically, parameters are sent through POST requests
  - Not visible to users from the URL

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
<html>
<body>

Welcome <?php echo $_GET["name"];
?><br>
Your email address is: <?php echo
$_GET["email"]; ?>

</body>
</html>
```

PHP – Handling Cookies

- Cookies are created from the server with the function *setcookie*
  - setcookie(*name, value, expire, path, domain, secure, httponly*);
  - Only *name* is the required parameter during creation
  - The expiration value is set in seconds
  - The *isset* function checks if specific parameters of the cookie are set

- Cookies can be accessed with the $_COOKIE variable

```php
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($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>
```

References

- HTML: https://www.w3schools.com/html/default.asp
- PHP: https://www.w3schools.com/php/default.asp
  - Online interpreter: http://sandbox.onlinephpfunctions.com/