

ICT286

Web and Mobile Computing

Topic 7

Server-Side Processing

With PHP

Objectives

- Understand how server-side scripting works and the difference between client-side scripting and server-side scripting
- Understand how to include PHP scripts in HTML. Understand the copy and interpret modes.
- Understand and be able to use Boolean, integer and float and string values and variables.
- Understand the differences between single quoted string literals and double quoted string literals.
- Understand type coercions and explicit type conversions.
- Understand and be able to use print, printf and echo to generate output.
- Understand and be able to write simple PHP code with conditions and control structures.

Objectives

- Understand and be able to send user input from HTML forms using HTTP GET method to server side.
- Understand and be able to send user input from HTML forms using HTTP POST method to server side.
- Be able to obtain client data using `$_GET` and `$_POST` arrays.
- Understand and be able to use strings and string related functions.
- Understand and be able to define and use new PHP functions.
- Understand and be able to handle files using PHP.

Readings

Sebesta: Chapter 9

Why Server-side Scripting?

- HTML is static.
- Client-side JavaScript is dynamic but only on the client-side
 - It can't get access to the resources on a server (e.g., a database)
 - There is certain information which we cannot be processed on the client (e.g., passwords).
- Server-side technologies, such as PHP, not only allow us to give users access to services, but also have the services processed by the server.

PHP

- Originally developed by Rasmus Lerdorf in 1994.
- Initially it stands for “Personal Home Page”. Later it became “PHP: Hypertext Preprocessor”
 - The definition is recursive, so don’t ask what the “PHP” in “PHP: Hypertext Preprocessor” stands for.
- PHP is a popular, general purpose scripting language that is especially suited to all kinds of dynamic web development and can be embedded into HTML.
 - Forms handling
 - Data exchanges between client and server
 - File processing
 - Database access

Why PHP?

- PHP enables the development of more dynamic web pages. PHP can verify forms, react to end-user input and access files and databases.
- PHP runs on the web server, so it can access files or databases on or connected to the web server.
- PHP is open source. It is available for free download on the Internet. There are many web pages and online tutorials that give information about PHP.
- PHP is an alternative to CGI, ASP.NET, Ruby, and JSP.

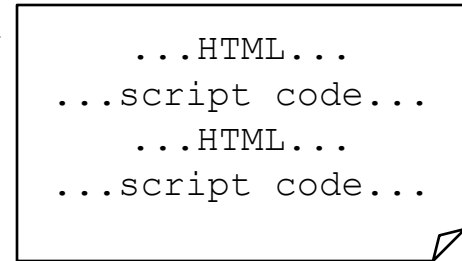
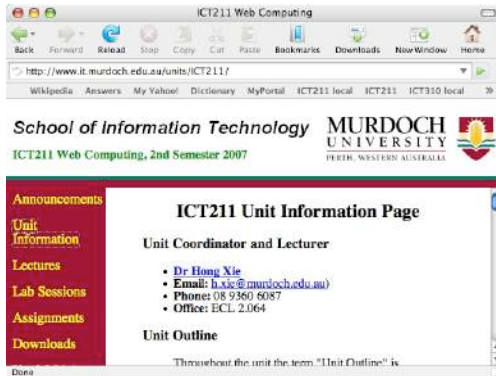
Basic Rules

- PHP syntax is similar to that of JavaScript
- PHP is interpreted just like JavaScript, but on the server-side.
- PHP is dynamically typed just like JavaScript
- PHP variable names are case sensitive, just like JavaScript.
- However, unlike JavaScript, keywords and function names in PHP are *case-insensitive*. However, you should adopt a consistent style in writing PHP to make it easier to follow.

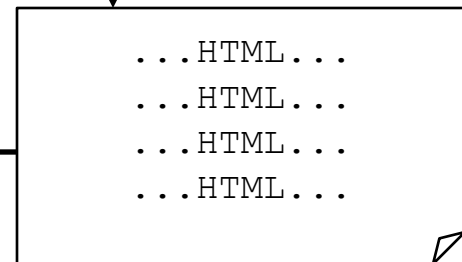
How Server-side Scripting Works

2. Web server finds requested file.

1. Browser requests URL



3. Web server invokes interpreter to turn the script code into HTML



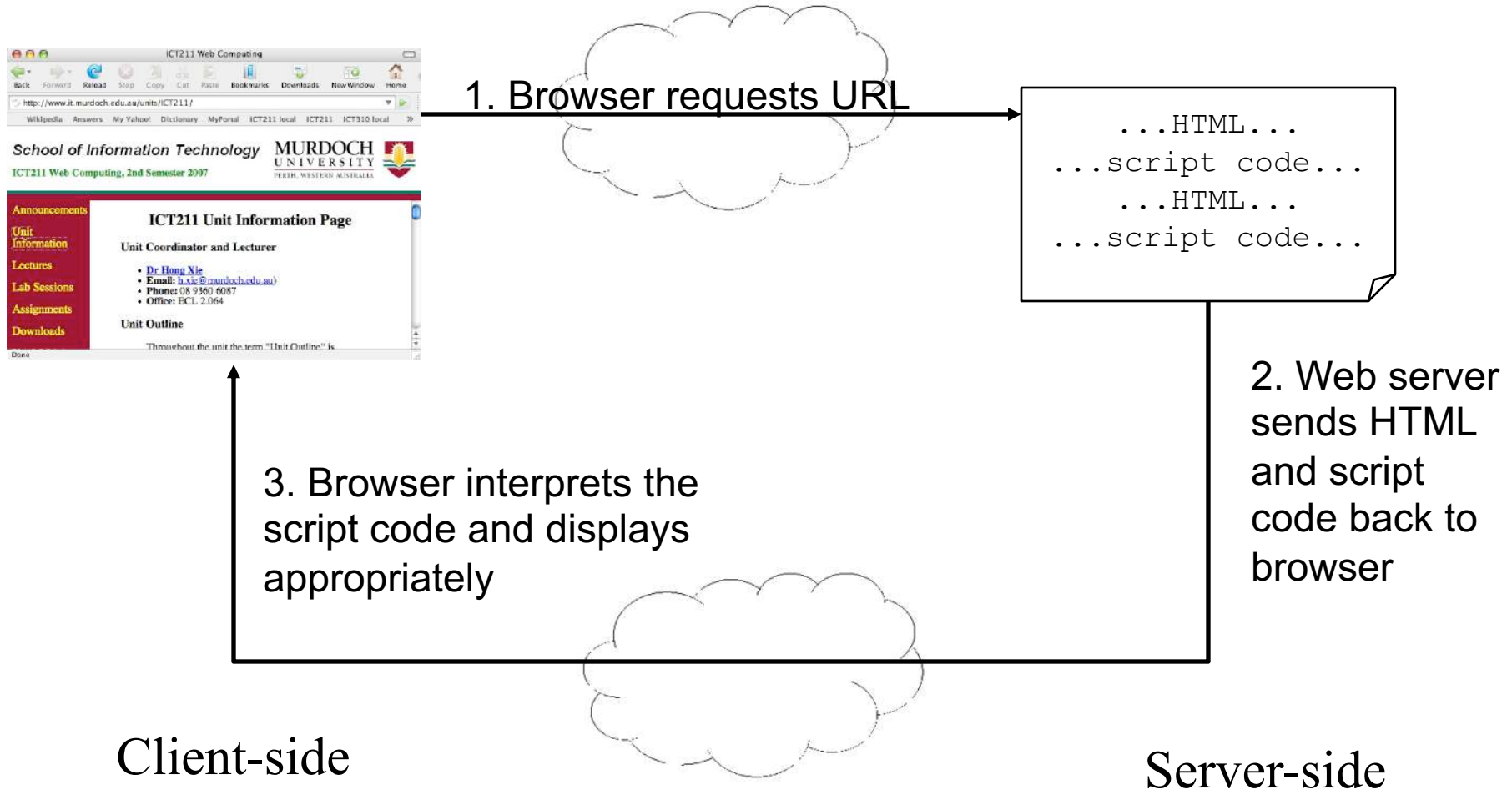
4. Web server sends HTML back to browser

5. Browser reads and displays HTML

Client-side

Server-side

Compared to Client-side Scripting

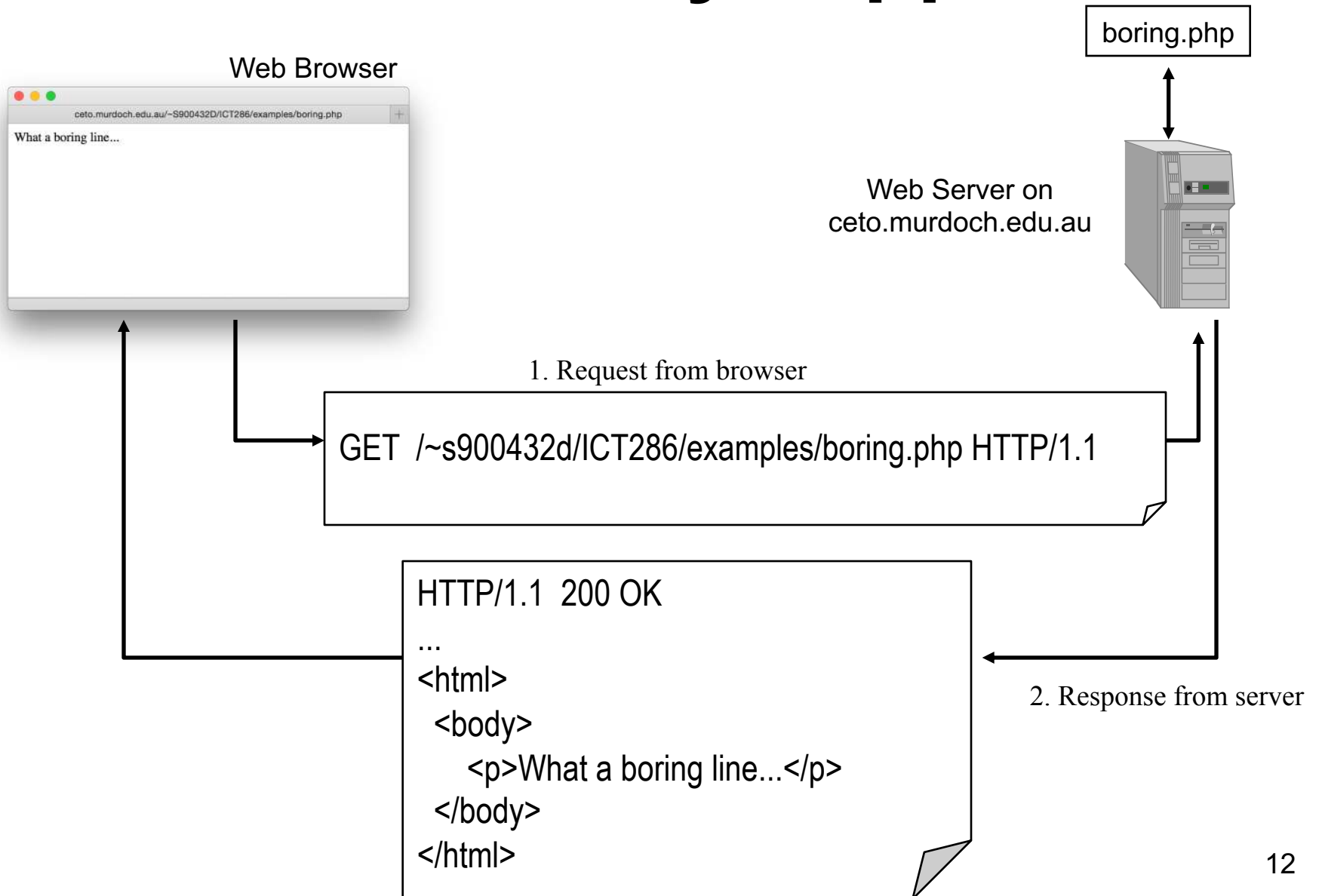


A Simple PHP Script

- All it does is print the words “What a boring line...” in a browser

```
<!DOCTYPE html>
<html>
<body>
    <?php
        echo "<p>What a boring line...</p>";
    ?>
</body>
</html>
```

What actually happens:



To Run PHP Scripts

- What it essentially involves is:
 - Create the script file using a text editor from the *public_html* directory under your home directory on ceto.
 - Access the script file using a browser and the appropriate URLs.

```
<!DOCTYPE html>
<html>
  <body>
    <?php
      echo "<p>Hello World</p>";
    ?>
  </body>
</html>
```

To Run PHP Scripts

- The output of the PHP script is a pure HTML code which is sent to the client for rendering.

```
<!DOCTYPE html>  
<html>  
<body>  
    <p>Hello World</p>  
</body>  
</html>
```

Include PHP Scripts

- Any PHP code must be enclosed within "<?php ...?>" tag.
- The PHP code can be embedded within the HTML code.
- It can also be read from an external file using `include` or `require` statements, such as

```
include 'table.php';  
require 'table.php';
```

- The file can also contain HTML code as well as PHP code. The PHP code must be enclosed within "<?php ... ?>" tag
- If there is an error reading the file, `include` will produce a warning, while `require` will produce a fatal error
- Web servers can be configured to recognise PHP files in different ways – e.g., by looking for file extension `.php`.

Copy or Interpret?

- The PHP processor has two modes:
 - Copy mode: if it encounters HTML code, it copies the code to the output
 - Interpret mode: if it encounters PHP code, it interpret the code and replace the PHP code with its output.

Comments

- Comments can be made within the script, and are enclosed within `/* ... */` brackets.
- or by placing a comment after a crosshatch sign (`#`)
- or after two forward slashes (`//`).
- Notice that comments can span several lines if the first method is used.
- All comments are ignored by the interpreter.

Variables

- All variable names in PHP begin with a dollar sign (\$) and must be followed by an underscore(_) or a letter, and then any number of letters, digits or underscores.
- Variable names in PHP are *case-sensitive*; `$MyString` is different to `$mystring` is different to `$myString` etc.
- You do not declare the data type of a variable.
 - The type is determined when it is assigned a value.
 - Thus the type changes during the execution of the program.

Unbound Variables

- In PHP, an unassigned variable has `NULL` value. An unassigned variable is also called an unbound variable.
- if you use an unbound variable, its `NULL` value may be coerced to another value depending on the context:
 - If the context is a number, it is coerced to `0`
 - If the context is a string, it is coerced to an empty string.
 - In a Boolean context, it is coerced to `False`
- You can test to see whether a variable is unbound using function `isset`, eg:
 - `isset($x)` returns `TRUE` if `$x` is assigned a value
 - `isset($x)` returns `FALSE` if `$x` is unbound

Primitive Types

- Four scalar types:
 - boolean
 - integer
 - float
 - string
- Two compound types:
 - array
 - object
- Two special types:
 - resource: points to an external resource such as a file or database connection
 - Null: only one value of null type: `NULL`

Example Basic Variable Types

```
<?php
```

```
$number = 5;           // integer type  
$real = 37.2;         // float type  
$done = TRUE;        // Boolean type  
$string1 = "this is a string"; // String type  
$string2 = 'this is another "string"';
```

```
?>
```

Boolean

- Boolean values are `TRUE` and `FALSE` (case insensitive)
- Non Boolean values can appear in Boolean context. These values will be evaluated to `TRUE` or `FALSE`:
 - Integer: to `FALSE` only if the value is `0`. Otherwise to `TRUE`
 - String: to `FALSE` only if the string is empty or `"0"`. Otherwise to `TRUE` (so string `"0.0"` is evaluated to `TRUE`!)
 - Double: to `FALSE` only if the value is exactly `0.0`.
 - Do not use double expression in the Boolean context

Integer and Double

- Numbers in PHP can be integer (integer type), for example 5, -12.
- or floating-point numbers (double type), i.e., with decimal point and/or exponent, for example, 3.14, .345, 345., 1.23e3.
- There are functions for numbers; for example:

```
$num2 = round($num1);
```

`$num2` will contain `$num1` rounded to the nearest integer

```
$num2 = round($num1, 2);
```

`$num2` will contain `$num1` rounded to 2 decimal places

```
$num2 = ceil($num1);
```

`$num2` will contain the smallest integer greater than or equal to `$num1`

Strings

- String literals can be quoted either with single quotes or double quotes, e.g.,

```
$s1 = "This is a string";
```

```
$s2 = 'This is another string';
```

- The dot (.) operator is used to concatenate two strings in PHP:

```
$fname = "Joe";
```

```
$lname = "Blow";
```

```
$name = $fname . " " . $lname;
```

`$name` **will contain** "Joe Blow"

Escape Sequences

- Like all languages, certain characters have special meanings. These special/control characters are called escape sequences. Some of the ones used in PHP:

Escape Sequence

Meaning

<code>\n</code>	insert a newline character
<code>\r</code>	carriage return
<code>\t</code>	horizontal tab
<code>\\</code>	backslash
<code>\\$</code>	dollar
<code>\"</code>	double quote

Single Quoted Strings

- In a single quoted string, escape sequences or variable names are not interpolated.

- Example 1:

```
$str = '"Good morning", he said.';
```

The string `$str` would contain

```
"Good morning", he said.
```

- Example 2:

```
$myName = "Hong";
```

```
$greeting = 'Hello, $myName\n';
```

The string `$greeting` would contain the following sequence of characters including the characters `\` and `n` at the end:

```
Hello, $myName\n
```

Double Quoted Strings

- If a double quoted string contains escape sequences or variable names, they are interpolated. E.g.,

```
$myName = "Hong";
```

```
$greeting = "Hello, $myName\n";
```

The string `$greeting` would contain the following sequence of characters

```
Hello, Hong
```

plus the new line character at the end.

- Note that you can do the same thing using:

```
$greeting = "Hello, " . $myName . "\n";
```

Operators

- These are very similar to JavaScript. Some examples:

<u>Operator</u>	<u>Meaning</u>	<u>Example Usage</u>
>	Greater than	2 > 3
!=	Not equal	\$x != 4
+	Addition	\$x + 2 + \$y
&&,	Logical AND, OR	(\$x > 4) && (\$y < 3)
++	Increment by one	\$y++ ;
.	Concatenation	\$x . " and " . \$y

etc

Type Coercion between String and Numbers

- When a number appears in the string context, it is implicitly converted to a string.
- When a string appears in the number context, it is implicitly converted to a number.
- If the string start with a sign or a digit, the string is converted
 - either to an integer value, if the string does not contain the decimal point or e or E
 - or to a float value, if the string contains decimal point or e or E.
 - Non numeric characters following the number in the string are ignored.
- If the string does not start with a sign or a digit, the string is converted to zero (integer 0).

Explicit Type Conversions

- Type casts, e.g.,
 - `(int)$total` or
 - `intval($total)` or
 - `settype($total, "integer")`
- The type of a variable can be determined with
 - `gettype`, e.g.,
 - `gettype($total)`
 - or `is_type`, e.g.,
 - `is_integer($total)`

Functions

- We can define and use functions:

```
<?php
    function add($a, $b) {
        return $a+$b ;
    }

    $value = add(2,3) ;
    print "<p>Two plus Three is $value.</p>" ;
?>
```

- Functions need not be defined before they are called.
- Functions can have a variable number of parameters.
- Function name is *not* case sensitive.

Some Notes on Functions

1. Use meaningful names for functions.
2. Use comments at the start of the function to describe it.
3. Always pass the correct number of arguments to the function.
4. Return a value from the function using the *return* statement.
5. By default, variables are passed by value, not by reference.
6. To pass a variable by reference put an ampersand (&) in front of the variable name – not recommended.

Predefined Functions

- To ensure that PHP reports all errors, set error reporting level to `E_ALL` at the beginning:

```
error_reporting (E_ALL);
```

- PHP has many predefined variables. You can find out these variables as well as the version of PHP running on the server and configuration information by calling function `phpinfo()` in a PHP script.

Dates and Times in PHP

- There are a set of functions you can use to get and format the date and time. For example:

```
echo (date("l dS F, Y"));
```

Will return the current date in the format:

```
Monday 16th September, 2019
```

- The function `getdate` will return an array containing all the parts of the current date and time.

```
$dates = getdate();
```

Strings Functions

Function	Meaning
<code>\$num = strlen(\$string);</code>	<code>\$num</code> will contain the number of characters in <code>\$string</code> .
<code>\$lower = strtolower(\$string);</code>	<code>\$lower</code> will contain <code>\$string</code> all in lower case.
<code>\$upper = strtoupper(\$string);</code>	<code>\$upper</code> will contain <code>\$string</code> all in upper case.
<code>\$new = ucfirst(\$string);</code>	<code>\$new</code> will contain <code>\$string</code> with the first character capitalized.
<code>\$new = ucwords(\$string);</code>	<code>\$new</code> will contain <code>\$string</code> with the first character of every word capitalized.
<code>\$new = trim(\$string);</code>	<code>\$new</code> will contain <code>\$string</code> with any spaces from the beginning or end of the string removed.
<code>\$new = chop(\$string);</code>	<code>\$new</code> will contain <code>\$string</code> with any spaces from the end of the string removed.
<code>\$part = substr(\$string, x, y);</code>	<code>\$part</code> will contain a sub-string of <code>\$string</code> starting from position <code>x</code> , for <code>y</code> characters.

Regular Expressions

- You will probably want to use Regular Expressions in your PHP code. The functions you will need are `preg_match`, `preg_match_all` and `preg_replace`.
- The rules you learned previously about constructing Regular Expressions are the same for PHP. The functions to use them are just slightly different. You can read about them in your text.

Produce Output

- Output from a PHP script is HTML (may also include JavaScript code) that is sent to the browser
- HTML is sent to the browser through the standard output
- We may use `print`, `printf` and `echo` to generate output. Examples:

```
print("<h2>Introduction</h2>");  
print 3.14;  
printf("The total is %d", $total);  
echo "<p>Hello", "World!</p>";
```

Produce Output

1. As PHP functions are case-insensitive; so `ECHO`, `echo` and `Echo` are all the same thing.
2. With `echo`, but not `print()`, you can send multiple, separate chunks of data using commas; e.g., `echo "Hello ", "World!";`
3. `echo` and `print()` can be used to print text over multiple lines.
4. You can use the *newline* character (`\n`) to provide line breaks in the HTML code to make it more readable.
5. To use a quotation mark in the HTML within the PHP, you *escape* it (`\"`).
6. The syntax of `printf` is the same as in C.

Program Control: Conditionals

- The `if` statement is similar to the `if` statement in JavaScript. Example:

```
if ($age >= 17) {  
    echo "You can take your driving test";  
}  
  
else {  
    echo "You are too young to take your  
driving test";  
}
```

- For all cases of the `if`, if there is only one statement, the curly brackets are not needed, but recommended.

Program Control: Conditionals

- Like in JavaScript, you can check multiple conditions:

```
if ($month == "April") {  
    echo "It is April";  
} else if ($month == "May") {  
    echo "It is May";  
} else {  
    echo "It is not April or May";  
}
```


Program Control: Conditionals

- The switch statement is nearly same as in JavaScript, and is used when a variable is to be checked against a number of values.

```
switch ($value) {  
    case "a":  
        echo "value is a\n";  
        break;  
    case "b":  
        echo "value is b<br />\n";  
        break;  
    default:  
        echo "the value is unknown<br />\n";  
        break;  
}
```

Program Control: Loops

- The loop-while loop will be executed at least once. This is because the test of the condition comes after the code has been executed. This is called a post-test loop.

```
do {  
    $c = test_something();  
} while ($c);
```

- If you want a loop that could (under certain conditions) execute zero times (i.e., not execute), then you need to use a different loop.

Program Control: Loops

- The while loop is the loop you need if there are conditions when the loop should not be executed. This is because the test of the condition comes before the code has been executed. This is called a pre-test loop.

```
while ($d) {  
    echo "ok<br />\n";  
    $d = test_something();  
}
```

Program Control: Loops

- The for loop is similar to that we encountered in JavaScript.

```
for ($i = 0; $i < 10; $i++) {  
    echo "The value of i is $i<br />\n";  
}
```

Program Control: Loops

- The foreach loop is used with arrays and we will examine it when we consider arrays later in this topic.

```
foreach ($numbers as $num) {  
    echo "$num <br />\n";  
}
```

Program Control: Be Careful!

- A condition can be true in PHP for a number of reasons. Apart from the usual of testing a Boolean value or a conditional statement

```
if ($a < 10) {  
    . . .  
}
```

- the following will return TRUE:

```
if ($var) {  
    . . .  
}
```

if `$var` has a value other than 0, an empty string, or `NULL`

Splitting Control Blocks

- You can split one control statement over multiple `<?php ... ?>` tags, e.g., the following are equivalent

```
<?php
    if ($a) {
        echo "<p>a is true.</p>" ;
    } else {
        echo "<p>a is not true.</p>" ;
    }
?>
```

```
<?php if ($a) { ?>
    <p>a is true.</p>
<?php } else { ?>
    <p>a is not true.</p>
<?php } ?>
```

Splitting Control Blocks

- Splitting one control statement over multiple `<?php ... ?>` tags is useful when you have a large chunk of HTML to print out, and do not want the hassle of putting them into multiple echo statements.

Processing HTML Forms

- One use of PHP is form processing.
- The form tag has two attributes:
 - action** : specifies the URL of the script to run on the server-side when the form is submitted.
 - method**: set to either *post* or *get*; defines the argument format used to send data to the script.
- If the form uses HTTP GET method, the user inputs from the form will be sent to the server script in the form of query strings.
- If the form uses HTTP POST method, the user inputs from the form are included in the body of the request message.

Obtaining Input Data from an HTML Form

- Any form input with a name attribute will be available in the PHP script from a global array.
- If the form uses HTTP GET method, the user input data will be available from the global array `$_GET`.
- If the form uses HTTP POST method, the user input data will be available from the global array `$_POST`.
- For example, if the form has a text box with a `name="UserName"` attribute, the PHP script that processes the form can get the user name either from `$_GET["UserName"]` or from `$_POST["UserName"]`, depending on which method is used by the form.

Example: Google Query Strings

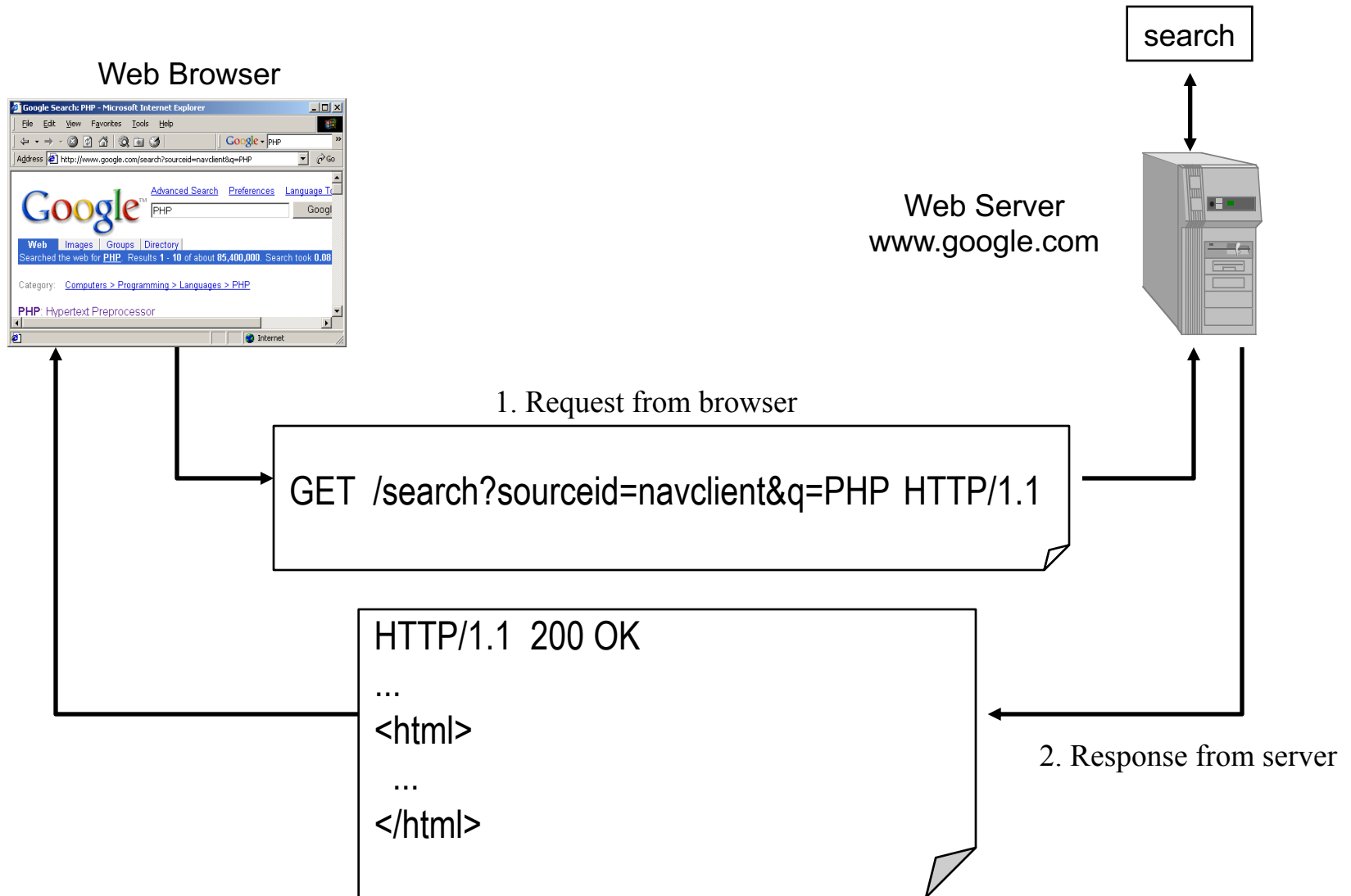


A lot of web applications use query strings to receive data from browsers.

Query String Format

- A query string of the format `"sourceid=navclient&q=PHP"` basically means:
 - There are two fields of data, separated by "&"
 - The first data field name is "sourceid", with value "navclient".
 - The second data field name is "q", with value "PHP".
- Data from HTML forms with method "GET" are sent to server scripts as query strings.

What Actually Happens:



Processing Forms that use GET method

```
<html>
  <body>
    <form method="GET"
      action="http://ceto.murdoch.edu.au/~s900432d/ICT286/examples/myname_get.php">
      <p> <label>Something to send: <input type="text" name="MyName" /> </label>
        <input type="submit" value="Submit" /> </p>
    </form>
  </body>
</html>
```

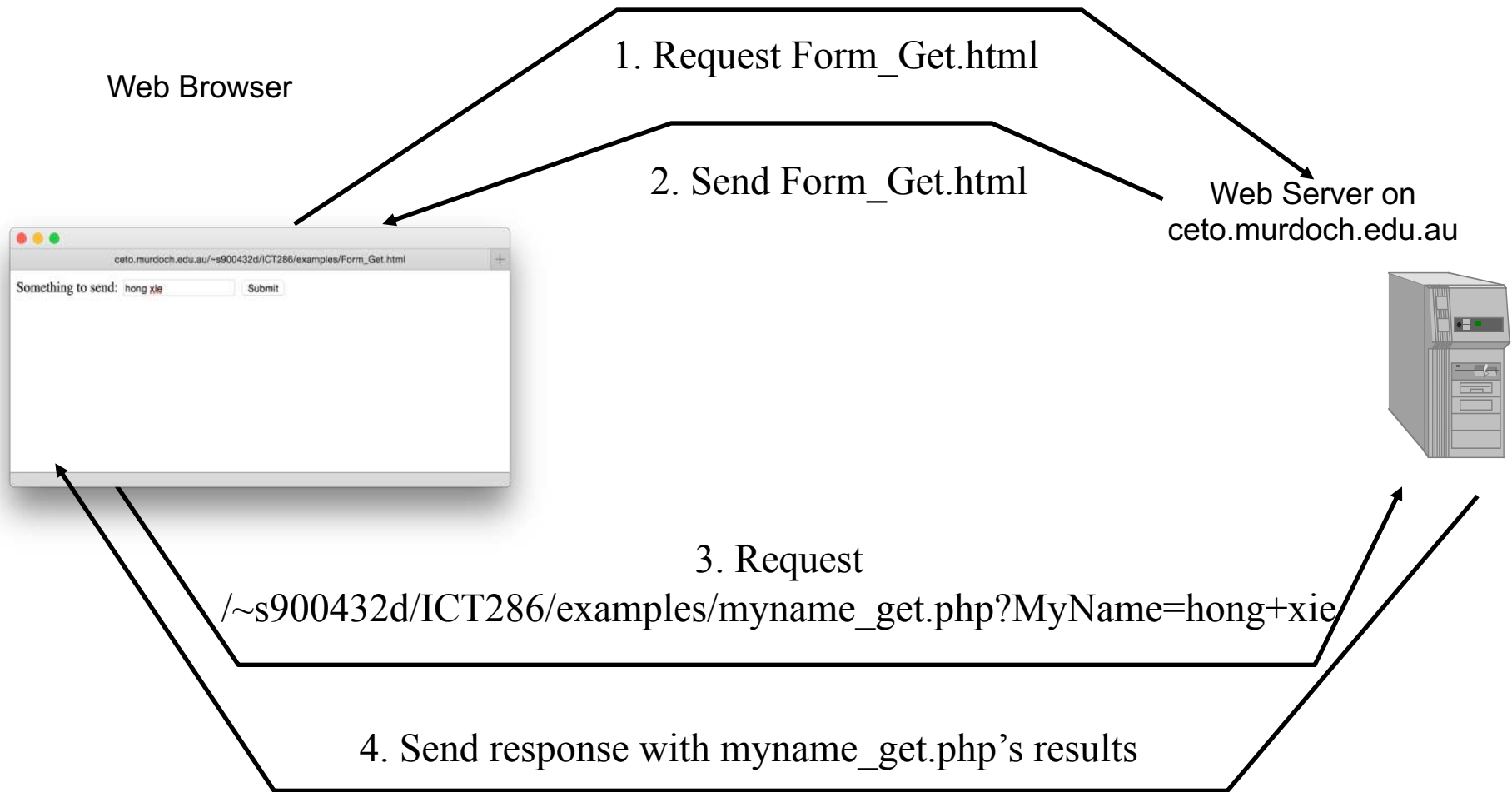
Form_Get.html

```
<html>
  <body>
    <p>
      <?php
        echo $_GET[ "MyName" ] ;
      ?>
    </p>
  </body>
</html>
```

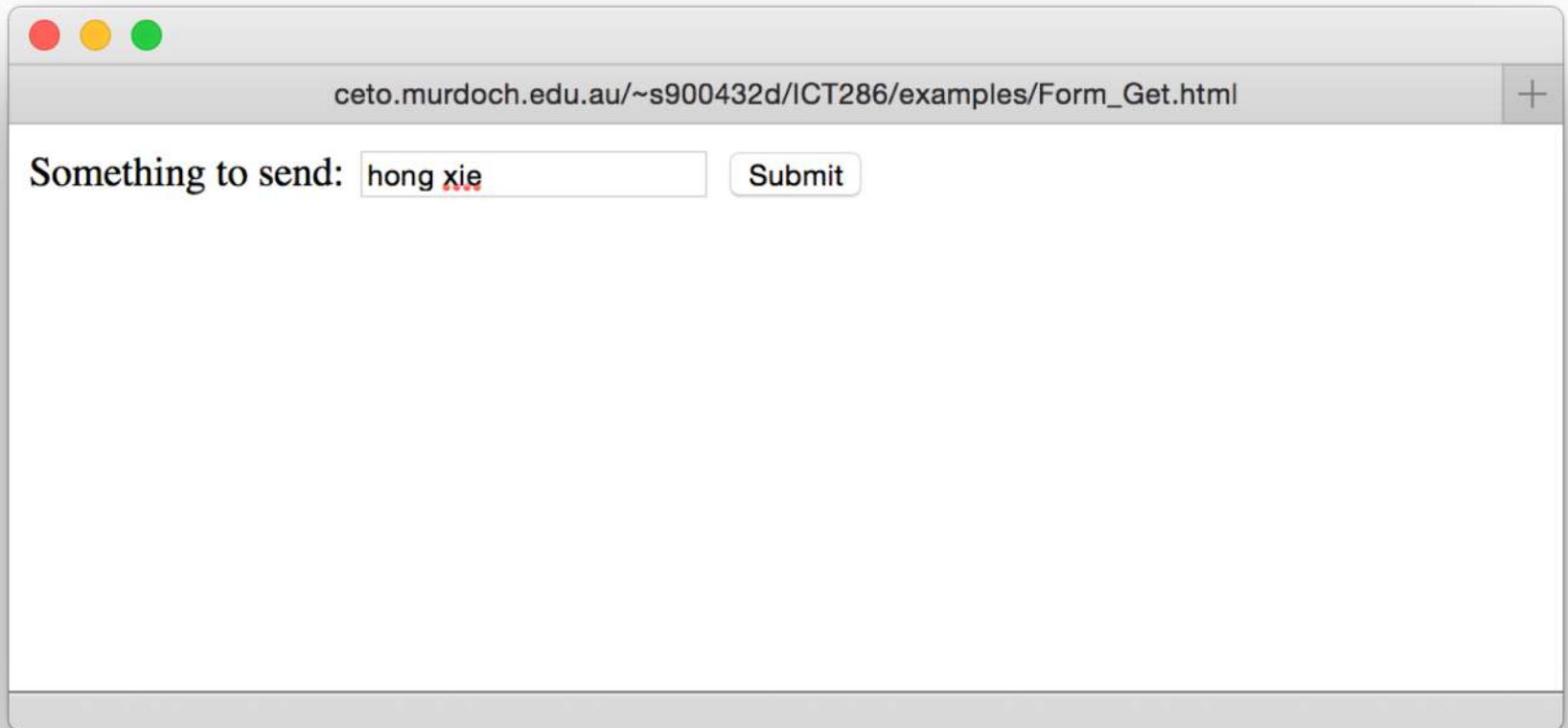
Myname_get.php

Note how the script has access to the form input element "MyName" through a global array `$_GET`.

Processing Forms that use GET method



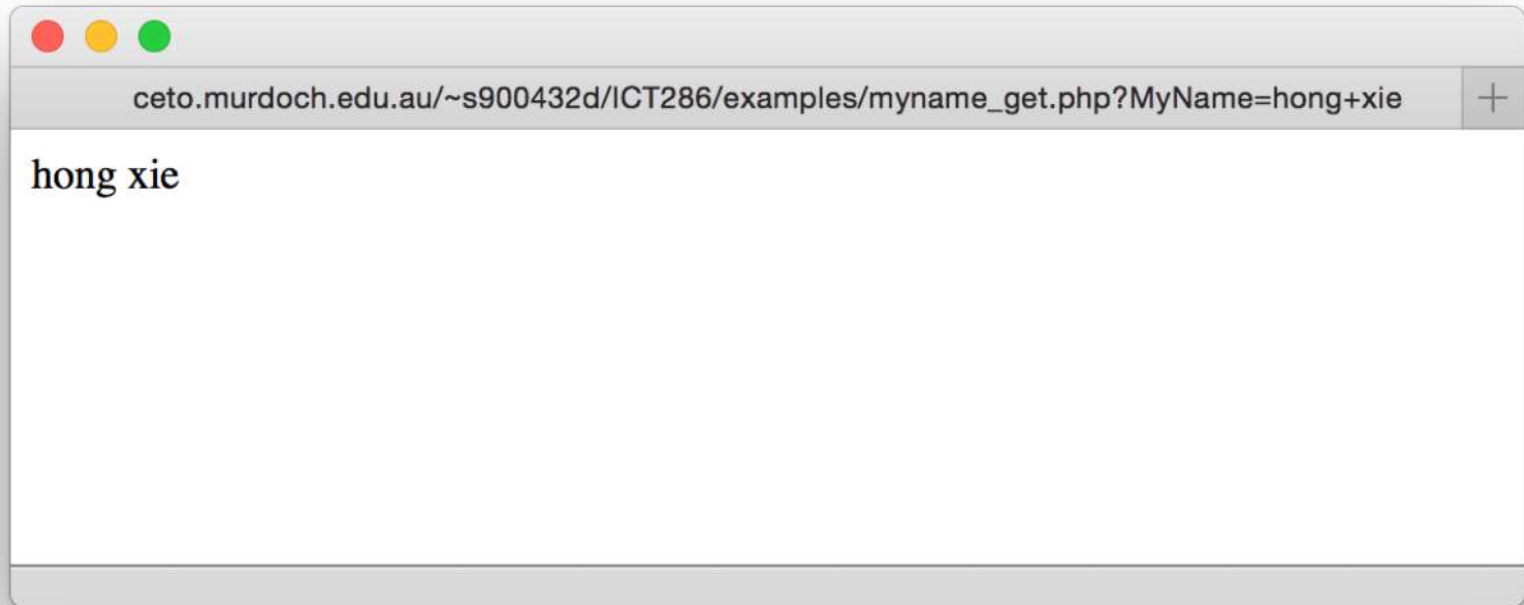
The Form Before Submission



A screenshot of a web browser window. The address bar shows the URL `ceto.murdoch.edu.au/~s900432d/ICT286/examples/Form_Get.html`. The main content area displays the text "Something to send:" followed by a text input field containing the text "hong xie". To the right of the input field is a "Submit" button. The browser window has a standard macOS-style title bar with red, yellow, and green window control buttons.

The page `Form_Get.html` displays a textbox for the user to type in something.

The Result after Pressing *Submit* Button



Note the new URL is pointing to the PHP script with the form input data appended to the URL in the form of a query string:

```
?MyName=hong+xie
```

Scripts and Forms in the Same directory

- If the PHP script is in the same directory as the HTML form document, we can put the relative path in the action attribute: `action="myname_get.php"` instead of the full URL

`action="http://ceto.murdoch.edu.au/~s900432d/ICT286/examples/myname_get.php"`

- This is better in cases when the web site consisting the script and form will be moved to different directories or different server together.
 - *Please use relative paths in your Major Assignments, as your files will be copied to an unknown directory for assessment.*

Scripts and Forms in the same directory

Note the change in the action field

```
<html>
<body>
  <form method="GET" action="myname_get.php">
    <p>
      <label> Something to send:
        <input type="text" name="MyName" />
      </label>
      <input type="submit" value="Submit" />
    </p>
  </form>
</body>
</html>
```

Form_Get.html

Using POST Method

```
<html>
<body>
  <form method="POST" action="myname_post.php">
    <p>
      <label> Something to send:
      <input type="text" name="MyName" /> </label>
      <input type="submit" value="Submit" />
    </p>
  </form>
</body>
</html>
```

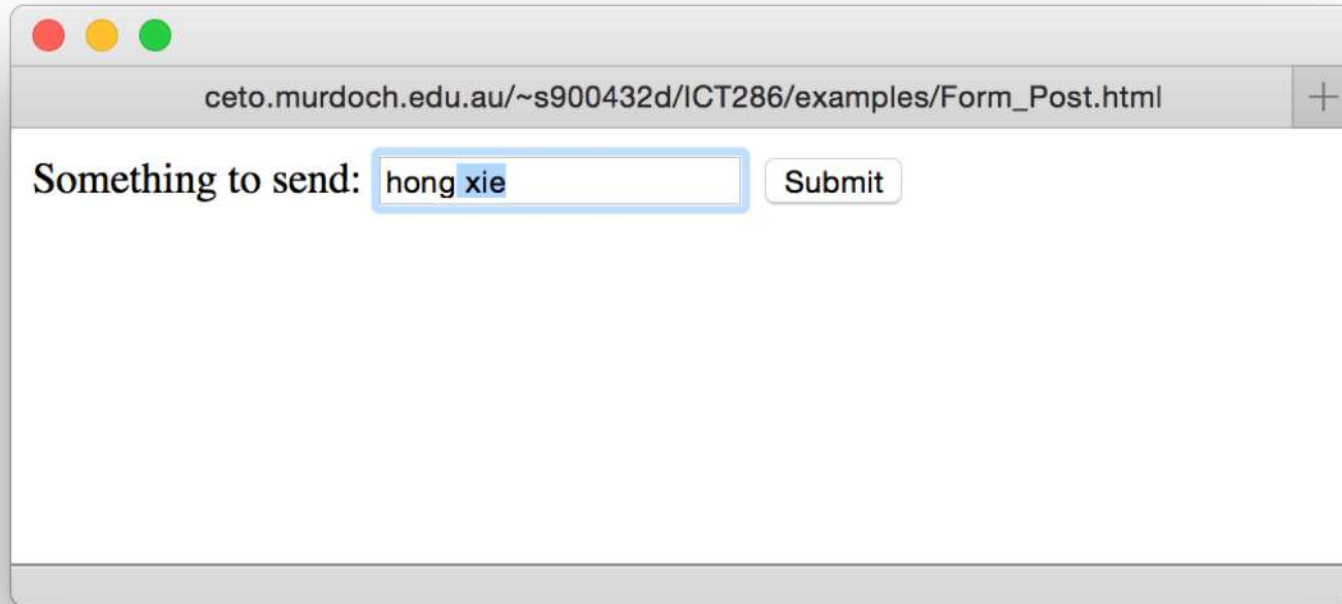
Form_Post.html

Using POST Method (cont'd)

```
<html>
  <body>
    <p>
      <?php
        echo $_POST["MyName"] ;
      ?>
    </p>
  </body>
</html>
```

Note that since the form uses POST method, the PHP script should use the global array `$_POST` to get the input value from the form.

The Form Before Submission



A screenshot of a web browser window. The address bar shows the URL `ceto.murdoch.edu.au/~s900432d/ICT286/examples/Form_Post.html`. The main content area displays the text "Something to send:" followed by a text input field containing the text "hong xie". The input field has a blue border and a blue highlight under the word "xie". To the right of the input field is a "Submit" button.

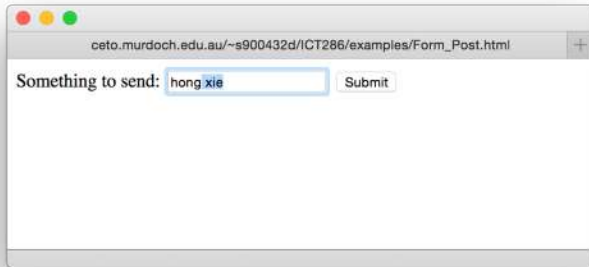
The Result After Pressing *Submit*



Note how the form data are not part of the URL, unlike `Form_Get.html`. This is useful for passing data you do not want to appear on the screen. NB: this does not ensure that the data is secure.

What Actually Happens:

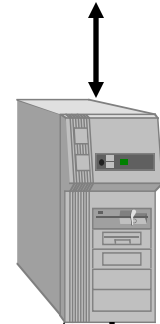
Web Browser



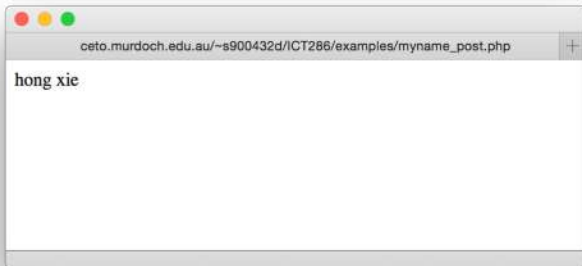
1. Request from browser, with data in the request body instead of the query string

```
POST /~s900432d/ICT286/examples/myname_post.php HTTP/1.1
...
MyName=hong xie
```

myname_post.php



Web Server



```
HTTP/1.1 200 OK
...
<html>
...
</html>
```

2. Response from server

File Handling

- Files are the simplest method for storing persistent data (i.e., data that do not disappear after the script finishes, unlike values in variables)
 - There are other, more sophisticated, ways (e.g., databases, cookies, etc.) but they involve more overhead or have storage limitations.
- PHP provides basic file-handling functions.

Example File Handling

```
$fd = fopen ("data.txt", "r"); # open data.txt for reading

while (!feof ($fd)) {      # while not reached end-of-file
    $buffer = fgets($fd);  # read one line into $buffer
    echo $buffer           # do something with $buffer
}

fclose ($fd);              # close the file
```

`$fd=fopen("data.txt","r")`

1. `$fd` is called the file handle and it 'points' at the file; this value is used when using the file, rather than the file name.
2. "data.txt" is the name of the file to be opened:
 - a. If the filename starts with HTTP, it is assumed to be on a remote web server and a standard HTTP session with the server is opened and the file is retrieved.
 - b. If the filename starts with FTP, it is assumed to be an ftp file and can be accessed. Note: if the file needs a user ID and password, these details can be included in the `fopen`.
 - c. If the file is in the current directory, just the file name is needed; otherwise the directory must be given. Use relative path for files.
 - d. If the file is on a Windows machine, the backslashes (`\`) in the path must be escaped; i.e., use two backslashes for each one required.

\$fd=fopen("data.txt", "r")

3. “r” is the mode to use when opening the file. The more common ones are:
- r opens the file for **reading**, starting from the beginning of the file.
 - r+ opens the file for **reading** and **writing**, starting from the beginning of the file.
 - w opens the file for **writing**, starting at the beginning of the file and sets the file length to zero; this means that any data in the file will be overwritten. If the file does not exist, it will be **created**.
 - a opens the file for **writing**, starting at the end of the file (**appending**). If the file does not exist, it will be **created**.

Note: you only have one copy of the file to read/write etc. This may have implications for the accuracy of the data; for example if someone else reads or writes the file while you are reading and writing it. If this is a problem, you may have to use a multi-user database.

`while (! feof ($fd))`

1. `feof ($fd)` returns `TRUE` if the end of the file is reached and `FALSE` otherwise.
2. Notice the use of the file handle `$fd`, not the name of the file.
3. The `while` statement processes each record within the file until the end of file is reached and then we will drop out of the loop. You will get an error if you try to read past the end of a file.

`$buffer=fgets ($fd) ;`

1. `fgets ($fd)` reads data from the file until a newline character is encountered, or end of file. In other words, one record at a time. The newline character is included as part of the string in `$buffer`.
2. Notice the use of the file handle `$fd`, not the name of the file.
3. If you are reading HTML, use `fgets ()`, which is the same as `fgets ()`, but will strip any HTML tags from the record.

`fclose ($fd) ;`

1. `fclose ($fd)` closes the file pointed at by the handle.
2. Notice the use of the handle `$fd`, not the name of the file.
3. You should always close files you have opened before ending the session.

Example File Handling

```
$fd = fopen ("data.txt", "a"); # open data.txt for appending
fputs($fd, "A new line."); # write a line into the end of file
fclose ($fd);             # close the file
```

1. This is an example of appending a new record to a file.
2. `fopen()` we have already considered. Here used with "a" for append.
3. `fputs($fd, "A new line.");` writes the string to the end of the file.
4. An optional length can be used with `fputs()`; in which case only that number of characters will be written.

Example File Handling

```
$fd = fopen ("data.txt", "w"); # open data.txt
                                # for overwriting
fputs($fd, "A new line.");
fclose ($fd);
```

1. In this example the existing file "data.txt" will be overwritten.
2. The file will be opened, the length of the file changed to zero, and the handle (`$fd`) will point at the beginning of the file, ready to write. If the file does not exist, one will be created, ready for writing. This may have implications, if you forget to include the directory, or incorrectly define the path to the file.
3. `fputs()` will write the string to the file and file "data.txt" will just contain that string when it is closed.

Other File Related Functions

- Use `file_exists (filename)` to determine whether file exists before trying to open it.
- Use `fread(file_handle, #bytes)` to read up to `#bytes` bytes from the file and returns it, or return `FALSE` if it encounters an error.
- Use `$bytes = fwrite(file_handle, string)` to write the string to the file.
- Files can be locked to avoid interference from concurrent accesses with `flock`.

References

- Main PHP site:
 - <http://www.php.net/>
 - A survey of other important PHP sites can be found there at <http://www.php.net/sites.php> and <http://www.php.net/links.php>.
- Tutorials:
 - <http://www.w3schools.com/php>