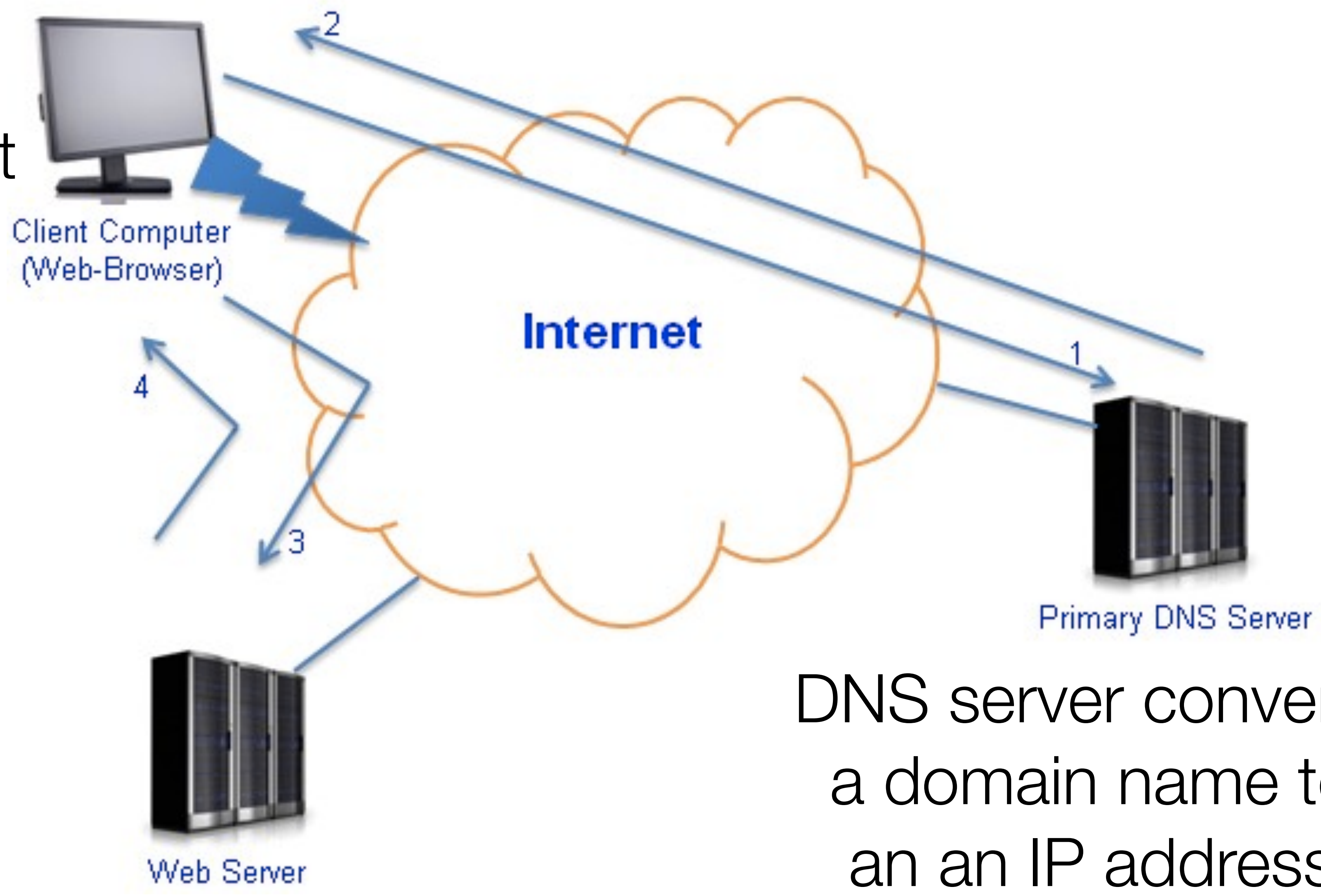
PHP: Web Programming

CS 377: Database Systems

World Wide Web (WWW)

Web browser is a program that receives web content and displays them



Web server serves content, whether it is webpages, images, movies, etc.

DNS server converts a domain name to an an IP address

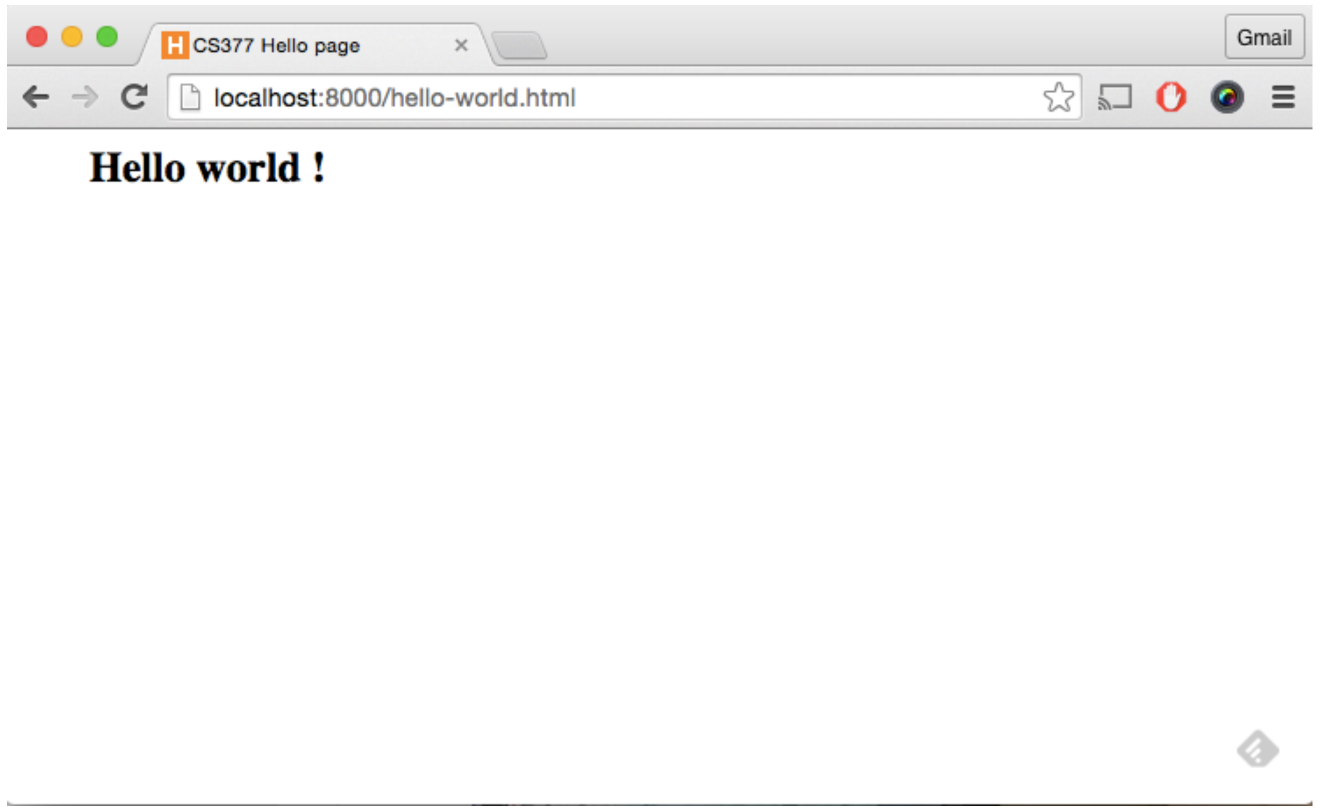
http://www.vebbsite.com/admin/photos/world-wide-web.jpg

Website

- Nothing more than a collection of computer files (webpages)
- Files are written in a special language called HTML (Hyper Text Markup Language)
    - Tags are used to specify how an items are displayed
    - Content can be static or dynamically generated

Example: Hello World HTML

```
<html>
   <head>
       <title>
           CS377 Hello page
       </title>
   </head>
   <body>
       <UL>
           <H2>
               Hello world !
           </H2>
       </UL>
    </body>
</html>
```

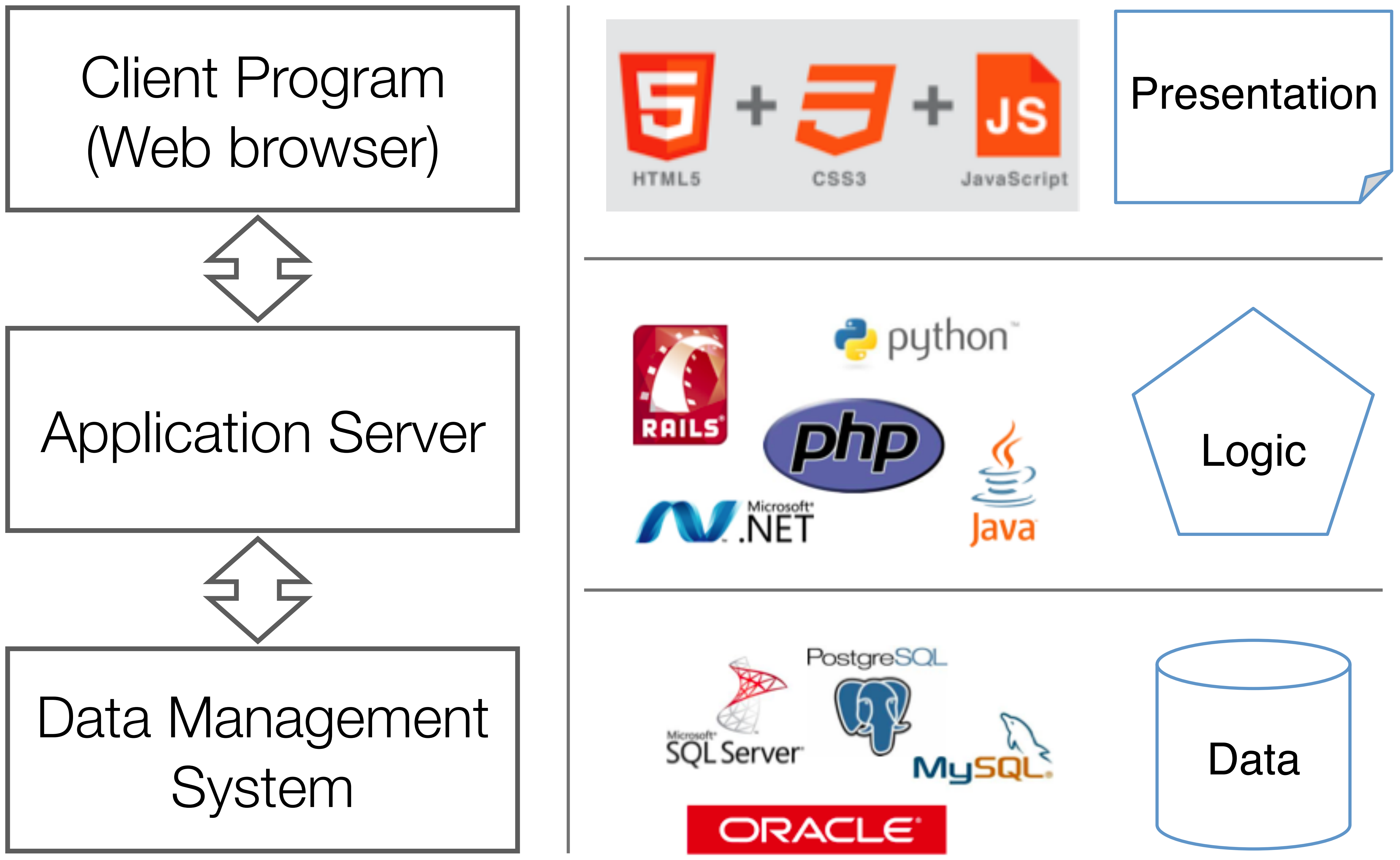


Hello world !

Dynamic Content

- Static webpages are considered passive content as they don’t perform any operations
    - Example: My personal webpage
- Webserver can execute programs that produce an HTML file (webpages)
    - Active content are web pages that are created dynamically
    - Common example is online ordering or shopping

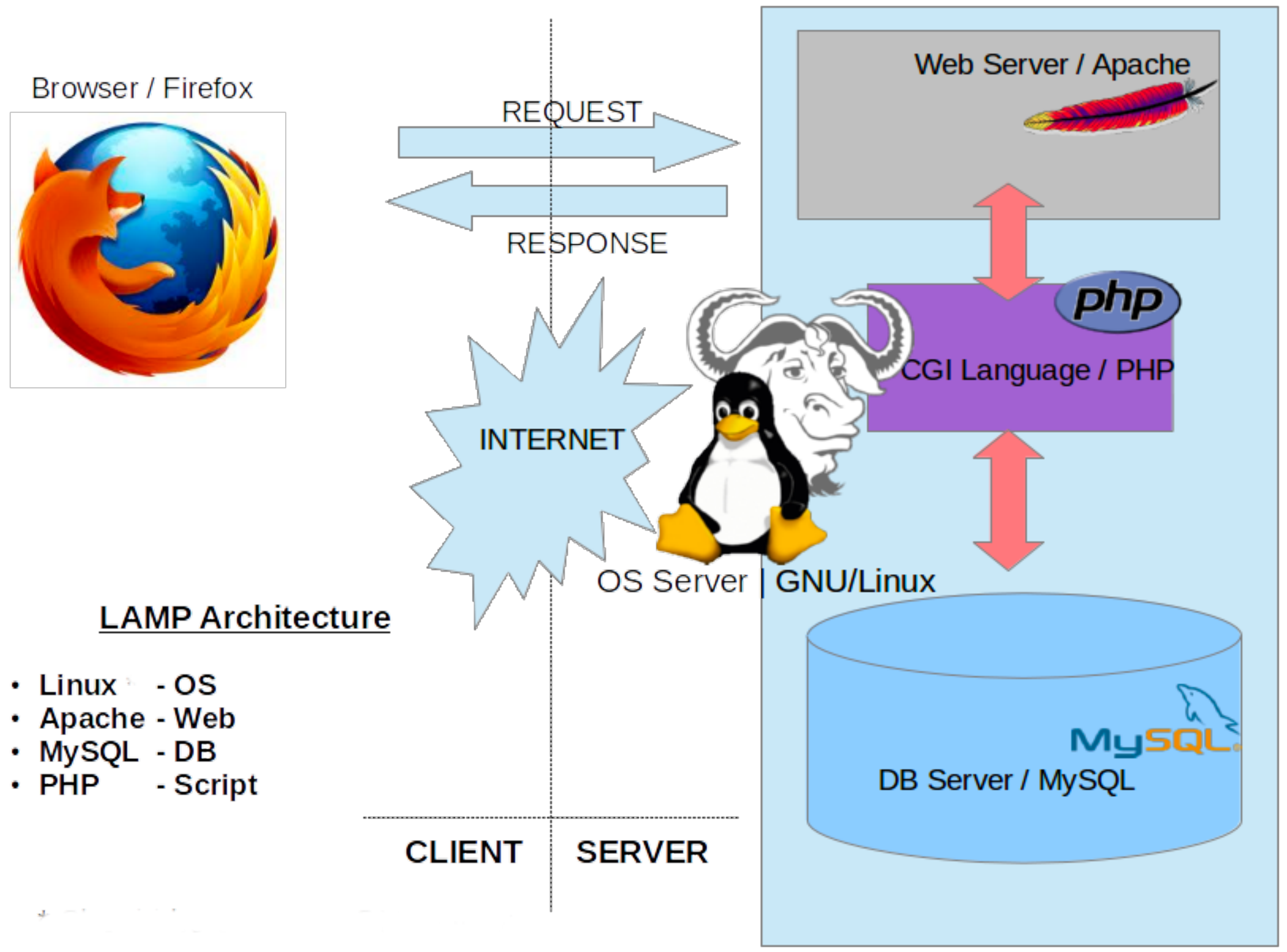
Three-Tier Web Architecture



http://apigee.com/about/blog/technology/new-three-tier-architecture-html5-proxy-apis

LAMP

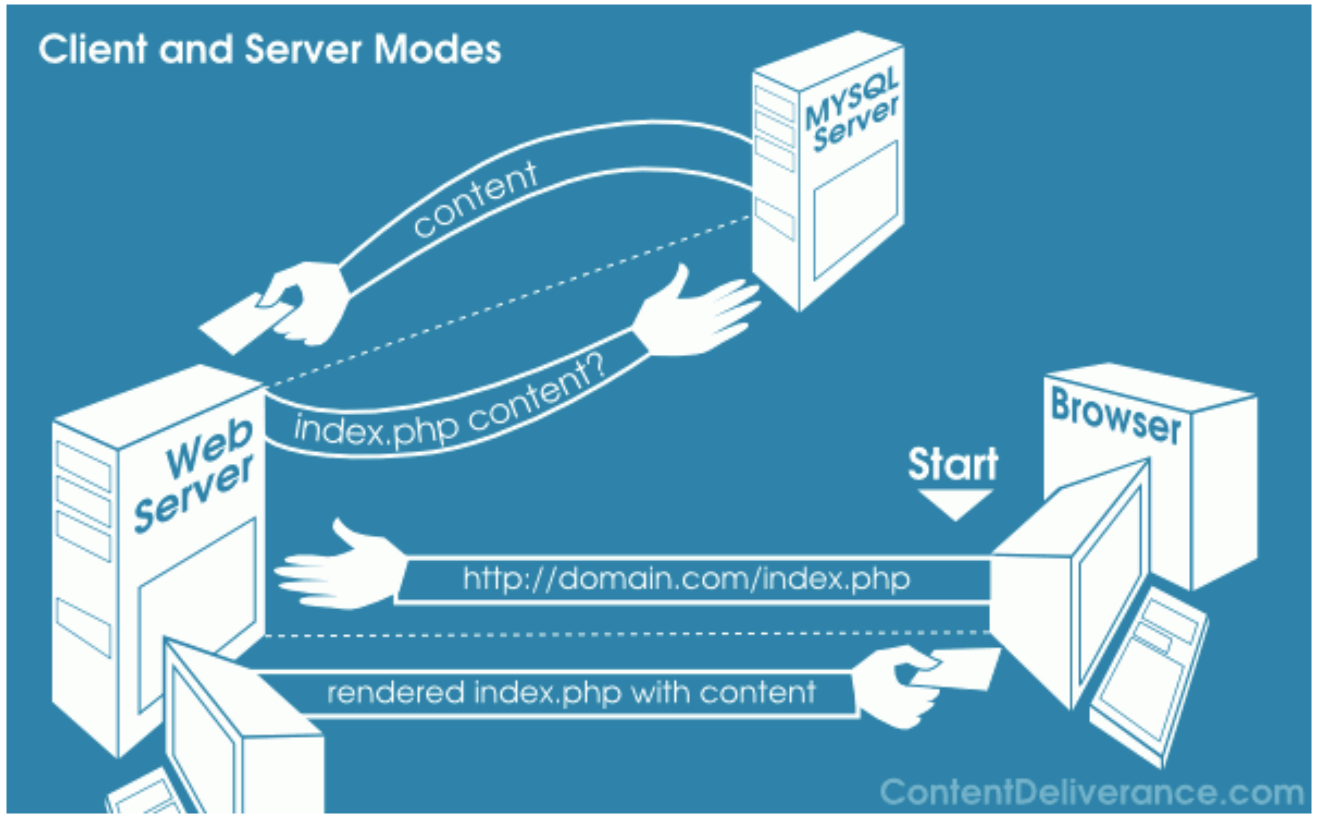
- Typical web service solution stack
- Original phrase was Linux, Apache, MySQL, and Perl (Perl —> PHP)
- Components are largely interchangeable



https://en.wikipedia.org/wiki/LAMP_(software_bundle)

PHP: PHP Hypertext Processor

- Open source, server-side scripting language for producing dynamic web pages
- Allows access to a database and executions of calculations and logic
- PHP web server interprets PHP code and dynamically constructs web page



http://contentdeliverance.com/cms-school/wp-content/uploads/2011/05/client-server-diagram-mysql.png

PHP: Strengths

- Ease of learning and use
- Open source and stable
- Speed - relatively fast
- Powerful library support & interface to many different database systems
- Availability of support

PHP: Disadvantages

- Security - many exploits of weaknesses of PHP
- Not suitable for large scales - not very modular
- Ugly and unpredictable type system (type casting and other conversion mechanism)
- Culture of messiness
- Poor debugging facilities

Working with PHP for CS377: Setup

- PHP is currently only installed on cs377spring16.mathcs.emory.edu
- Created a username (your netID) on the machine with the password: <your netID> + # + <studentID>
- Remote login to the server:
  ssh -X <username>@cs377spring16.mathcs.emory.edu
- Work should be done inside your public_html directory:
  cd ~/public_html
- You can access your PHP scripts via a brower:
  http://cs377spring16.mathcs.emory.edu/~<netID>/filename

PHP Program Structure

- PHP code can be embedded in HTML code
- PHP program consists of
    - Main program
        - Statements enclosed by the PHP tags
    - Function definitions

PHP Interpreter

- Echo everything that is not enclosed inside a PHP tag
- Text that is enclosed inside a PHP tag are considered to be PHP
  - Syntax:
    <? php
    … one or more PHP statements …
    ?>
- Example file without any PHP tags:
  This is just a text file without a PHP tag. It just repeats exactly what I've typed.

Example: Hello World in PHP (helloworld.php)

```
<?php
// prints hello world
echo "Hello World!";
?>
```

start tag to denote PHP statement

comment

end tag to close PHP statement

http://cs377spring16.mathcs.emory.edu/helloworld.php

Running PHP Programs

- Stand-alone (good for debugging)
  - UNIX-prompt>> php <script-name>
- Web browser
  - PHP script inside ~/public_html on cs377spring16 server
  - Point your favorite web browser to: http://cs377spring16.mathcs.emory.edu/~<userid>/<script-name>

Example: PHP with HTML (luckyNum.php)

```
<html>
<head>
<title> PHP Test </title>
</head>
<body>

<UL>
Welcome stranger, here is your lucky number:
<?php print rand(1, 1000); ?>
</UL>

</body>
</html>
```

http://cs377spring16.mathcs.emory.edu/luckyNum.php

PHP Variables

- Syntax: $variableName
- Variables start with letter or underscore
- Variable name is case-sensitive
- Implicitly defined — automatically defined when you use the variable for the first time in a program

Example: PHP Variables (var.php)

```
<?php
  $a = 1;
  $A = 2;
  print("a = " . $a . "\n");    # . is string concatenation
  print("A = " . $A . "\n");   # Var name is case sensitive !
  print("b = " . $b . "\n");   # Warning, not fatal !
?>
```

http://cs377spring16.mathcs.emory.edu/var.php

PHP Variable Types

- Support 8 primitive types
    - 4 scalar types: boolean, integer, float, string
    - 2 compound types: array, object (C's struct)
    - 2 special types: resource (special variable holding a reference to an external resource), NULL
- Dynamic typing — type of variable is determined by the type of value that was stored in the most recent assignment statement

Example: Dynamic Typing (dynatype.php)

```
<?php
  $a = 12;
  print ("a = " . $a . " Type of a = " . gettype($a) . "\n");
  $a = 12.0;
  print ("a = " . $a . " Type of a = " . gettype($a) . "\n");
  $a = "12";
  print ("a = " . $a . " Type of a = " . gettype($a) . "\n");
  $a = true;
  print ("a = " . $a . " Type of a = " . gettype($a) . "\n");
?>
```

PHP Operators

- Operators are similar to Java
  - Arithmetic operators: +, -, *, / , %, **
  - Logical operators: and, or, xor, !
  - Comparison: ==, !=, <, <=, >, <=
- Example:

```
<?php
  $b = 3 * 3 % 5;
  $b = $a++ + 23;
  $a = ++$b - 23;
?>
```

PHP Statements

- If statement & elsif
  if (expr1)
  {
  .. statements …
  }
  elseif (expr2)
  {
  .. statements 2…
  }
  [else
  { .. more statements …
  }]

- While statement
  while (expr)
  {
  .. statements …
  }

- For statement
  for (expr1; expr2; expr3)
  {
  … statements …
  }

break and continue work similarly

PHP Functions

- Similar to functions in other programming languages
- Can appear anywhere in the main program
- Need not be defined before it is used
- Syntax:
  function <funcName> ($<param1>, $<param2>, …)
  {
  … one or more statements …
  }

Example: PHP Function (func1.php)

```
<?php
   $a = square(4);
   print("Square of 4 = " . $a . "\n");
   # Function definition
   function  square( $x )
   {
      $r = $x * $x ;
      return( $r );
   }
?>
```

prints out the 2 of 4 = 16

PHP Variable Scope

2 scopes in PHP

- Global (program) scope — variable created in the main program has global scope and can be accessible from everywhere in the main program
    - Access variable inside a function by declaring it a global variable with the keyword global
- Function scope — variable created in the function has a function scope and will be different than a variable with the same name in global scope

Example: PHP Variable Scope (varscope.php)

```
<?php
 $a = 1;              # Global a
 print("Main: a = " . $a . "\n");
 f($a);
 print("Main: a = " . $a . "\n");

 function  f( )
 {
    global $a;      # ******** a will now access a global variable
    print("f before: a = " . $a . "\n");  # Global scope a
    $a = 4444;
    print("f before: a = " . $a . "\n");  # Global scope a
 }
 print("Main: a = " . $a . "\n");
?>
```

Beware PHP Weirdness

- Things in PHP that unlike Java/C
  - String — different ways to quote a string
    - Variables can appear inside a string
    - Variables are evaluated differently depending on how the string is quoted
  - Array — use of associative arrays (key, value pairs)

Strings

- Single-quote strings — always treated verbatim and no evaluation takes place
  - Example:
    $x = 1;
    print ‘This is a single-quoted string. This is $x\n’;

    Output:
    This is a single-quote string. This is $x\n

Strings (2)

- Double-quote strings — perform evaluation of variables to construct final strings
  - Use escape character “\” before $ to prevent evaluation
  - Example:

```
$x = 1;
print “This is a double-quoted string. This is $x\n”;
print “This is an escaped double-quoted string. This is \$x\n”;
```

    Output:

```
This is a double-quote string. This is 1
This is an escaped double-quote string. This is $x
```

Strings (3)

- “Here” documents— inline multi-line text that is evaluated
  - Example:

```
$x = 12345;
print <<<MARKER
    Here document text.. type away... This is $x
    Another line. Just keep going - the string will not stop
    until there is a line with MARKER at the START of the line\n
MARKER;
```

Output:

```
Here document text.. type away... This is 12345
Another line. Just keep going - the string will not stop
until there is a line with MARKER at the START of the line
```

Example: Strings (String1.php)

```
<?php
 $x = "Hello World !";
 print 'Single-quoted string. This is $x';
 print"\n";
 print "Double-quoted string. This is $x";
 print"\n";
 print <<<MARKER
  Here document text.. type away... This is $x
  Another line. Just keep going - until a line with MARKER is found
MARKER;
 print"\n";
 print <<<MARKER2
  Here document text.. type away... This is \$x
  Another line. Just keep going - until a line with MARKER is found
MARKER2;
 print "\n";
?>
```

Arrays

- Array is an ordered map — associates keys with values
  - General:
    ```
    $varName = array (
                       key1 => value1 ,
                       key2 => value2 ,
                       ...
                      );
    ```
  - Integer indices
    ```
    $varName = array (
                       value1 ,
                       value2 ,
                       ...
                      );
    ```
  - “Traditional” way
    ```
    $arrName[ index ] = value;
    ```

Array Functions

- Count the number of elements in an array:
  count($<array variable>)

- Accessing elements in array

```
foreach ($<array variable> as $KEY_VAR =>
$VALUE_VAR)
{
    $KEY_VAR = key of the current array element
    $VALUE_VAR = value of current array element
}
```

Example: Associate Arrays

- array01.php — different syntax for defining an array
- array02.php — counting the number of elements in an array
- array03.php —accessing the array using the special foreach structure
- array04.php — an example of a true associate array where the keys are not integers

PHP Program Steps

- Connect to the database
- Execute a query to get a result set
- Use a loop statement to obtain result tuples from result set
- Free resources
- Disconnect

Looks very similar to JDBC program…

Step 1: Connecting to Database

Access to MySQL database can be done with

- ext/mysql (MySQL extension which is not recommended and deprecated now)
- **ext/mysqli (MySQL improved extension)**
- PDO (PHP Data objects - pure object oriented programming)

Step 1: Connecting to Database (2)

Connect to MySQL database server using mysqli_connect() function

- Syntax: mysqli_connect (host, user, passwd [, dname [, port [, socket]]])
- Example:

```
$conn = mysqli_connect("cs377spring16.mathcs.emory.edu",
"cs377", "abc123");
 // check connection
 if (mysqli_connect_errno())
 {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
 }
```

Step 1: Connecting to Database (3)

- Specify the database in the connection:
  $conn = mysqli_connect("cs377spring16.mathcs.emory.edu","cs377","abc123", "companyDB");

- Use **mysqli_select_db()** function:

```
if ( ! mysqli_select_db ($conn, "companyDB") )
{
      printf("Error: %s\n", mysqli_error($conn) );
      exit(1);
}
```

Step 2: Submitting a SQL Query

- Execute a query using mysqli_query()

```
if ( ( $result = mysqli_query( $conn, "SQL-
command" ) ) == 0 )
{
    printf("Error: %s\n", mysqli_error($conn));
    exit(1);
 }
```

- Returns 0 if there was an error, otherwise the result

Example: Submitting a SQL Query

- SQL query:
  SELECT fname, name, salary FROM employee;

- PHP code:

```
$conn = mysqli_connect(“cs377spring16.mathcs.emory.edu”,”cs377",
"abc123", "companyDB");
 if (mysqli_connect_errno())
 {
      printf("Connect failed: %s\n", mysqli_connect_error());
      exit(1);
 }
 $query = 'select fname, lname, salary from employee';
 if ( ! ( $result = mysqli_query($conn, $query)) )
 {
     printf("Error: %s\n", mysqli_error($conn));
     exit(1);
 }
```

Step 3: Obtain SQL Results

Many different functions to retrieve result tuples

- **mysqli_fetch_all($result)** : fetches all result rows and returns the result set as an associative array
- **mysqli_fetch_array($result)** : returns the current (fetched) row as an array
- **mysqli_fetch_assoc($result)** : returns the current (fetched) row as an associative array or NULL if there is no more rows

Step 3: Obtain SQL Results (2)

- Focus on **mysqli_fetch_assoc($result)**
- Returns associative array that contains (key, value) pairs with the attribute name and value
- Example:

$key	$value
SSN	111-11-111
Fname	John
Lname	Smith
…	…

Example: Print SQL Results

Print attribute names and attribute values from $result array

```
while ( $row = mysqli_fetch_assoc( $result ) )
{
     foreach ($row as $key => $value)
     {
     print ($key . " = " . $value . "\n");
     }
     print("==================");
 }
```

Example program: employee0.php

Step 4: Free Resources & Disconnect

- De-allocate and free resources using mysqli_free_result()
    - Syntax: **mysqli_free_result(<result variable>);**
- Disconnect our connection with MySQL server using mysqli_close()
    - Syntax: **mysqli_close(<connection variable>);**

Example: Stand-Alone PHP program

- Print all the employees in the company database in a "tabular" format
- Print the attribute names only once
- Print the tuples
- To RUN: PHP emp-table.php

PHP via Web Browser

- Extremely easy to execute a program with a web browser
- Add some HTML header and trailer tags to the PHP script
- Put the MySQL PHP script in the special directory (will depend on what web server architecture you use)
- Load the PHP script in the web browser

Example: PHP Program via HTML

- HTML is ideally suited for formatting outputs
- Same example as before where you want to display all the employees in the company database in a “tabular” format — utilize HTML table format
    - <TABLE> tag to denote start of table
    - <TR> denotes a new row
    - <TD> denotes one data item in the row
- Example: emp-html-table.php

http://cs377spring16.mathcs.emory.edu/emp-html-table.php

PHP: Obtain User Input via FORM

HTML FORM tag allows a webpage to obtain input field(s) from the user

- <input> element
  - <input type = “text”> defines a one-line input field for text input
  - <input type = “radio”> defines a radio button (limit to 1 choice)
  - <input type = “submit”> defines button to submit a form to form-handler

PHP: Obtain User Input via FORM (2)

- <input> element
  - Each input field must have a name attribute
    <input type="text" name="varname">
  - Optional: specify the size of the input filed
    <input type="text" name="varname", size=40>

PHP: Obtain User Input via FORM (3)

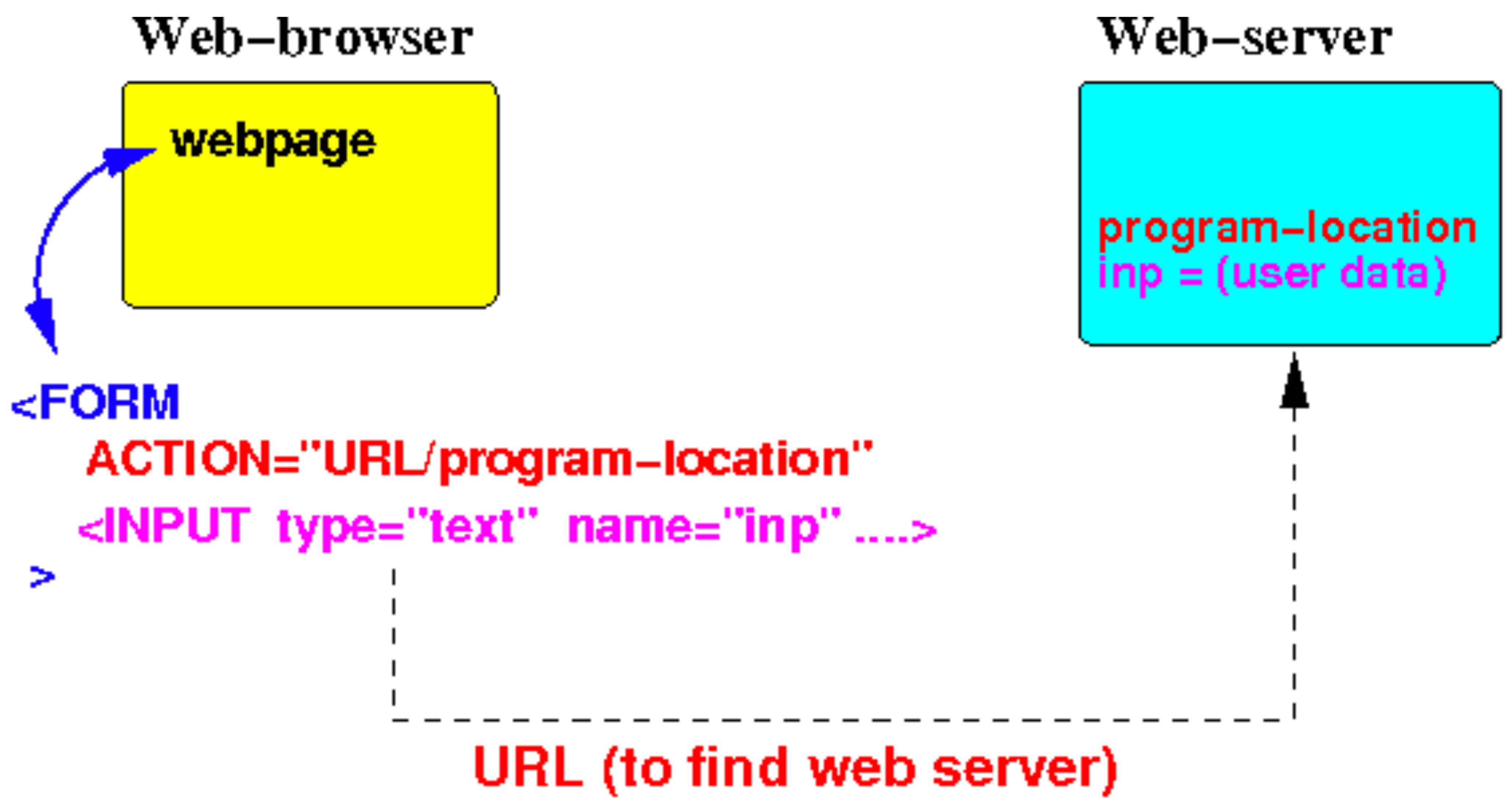
- <form action="filename.php" method="{get | post}">
  - action defines the address or URL where to submit the form
  - method specifies the HTTP method to be used when submitting the forms
    - GET (default) is generally used for short amounts of data and without sensitive information (data is encoded after a ? symbol)
    - POST offers better security because submitted data is not visible in the page address

Example: HTML FORM (form1.html)

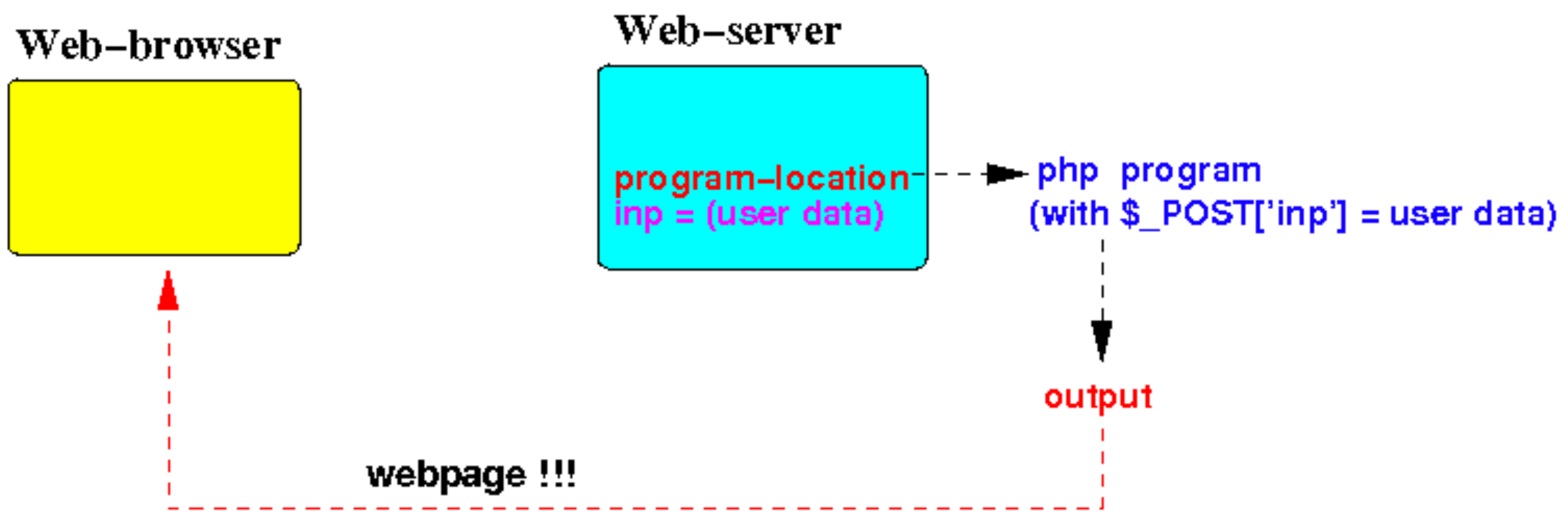
```
<html>
<head>
   <title> HTML Form 1</title>
</head>
<body>
   <HR>
   <HR>
   <B> Form: </B>
   <HR><P>
   <FORM ACTION="http://cs377spring16.mathcs.emory.edu/echo.php"
METHOD="POST">
      <p>Enter input: <input type="text" name="inp" size=40></p>
      <p><input type="submit" value="Press to send"></p>
   </FORM>
</body>
</html>
```

http://cs377spring16.mathcs.emory.edu/form1.html

PHP: Receiving Data using POST



PHP: Receiving Data using POST (2)



PHP: Receiving Data using POST (3)

- PHP interpreter receives input fields in a form tag in a web page
    - Associative array named $_POST[]
    - Initializes the element $_POST[‘input-var-name’] with the value entered in the corresponding input field in the form tag

Example: PHP Script for form1 (echo.php)

```
<html>
<head>
<title> Form1 test </title>
</head>
<body>
<HR>
<B>
<?php
# -----------------------------------------------------------------
# PHP program: echo the data send in the "inp" field by the form
# -----------------------------------------------------------------
   $data = $_POST['inp'] ;
   print("Post Data is $data \n");
?>
</B>
<HR>
</body>
</html>
```

Example: PHP Client for companyDB

- Web form to submit a query: http://cs377spring16.mathcs.emory.edu/companyDB-queryform.html
- PHP script to handle the query: http://cs377spring16.mathcs.emory.edu/companydb-query.php

PHP: Recap

- How to serve dynamic content on the web
- PHP basics
- PHP with MySQL program steps
- HTML web forms w/ PHP
- For more information about PHP: http://us2.php.net/manual/en/index.php