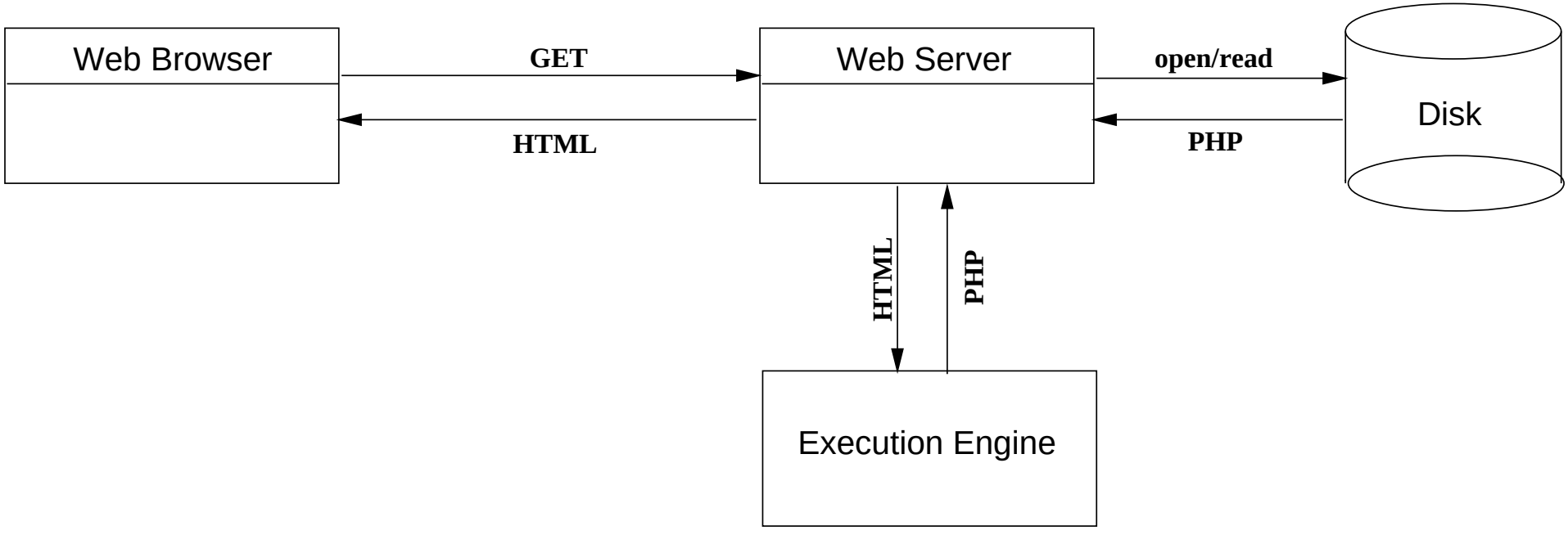
The Syllabus' Actual State

- Introduction to the client/server computing paradigm
- The HTML standard
- Cascading Stylesheets (CSS)
- The scripting language JavaScript and DOM
- An in-depth coverage of PHP programming techniques
- The MySQL backend databases and PHP
- Security aspects in client/server systems
- Distributed objects, RMI, CORBA

The Syllabus' Actual State

- Introduction to the client/server computing paradigm
- The HTML standard
- Cascading Stylesheets (CSS)
- The scripting language JavaScript and DOM
- An in-depth coverage of PHP programming techniques
- The MySQL backend databases and PHP
- Security aspects in client/server systems
- Distributed objects, RMI, CORBA

Architecture of the c/s PHP



What is PHP?

A simple yet powerful language designed for:

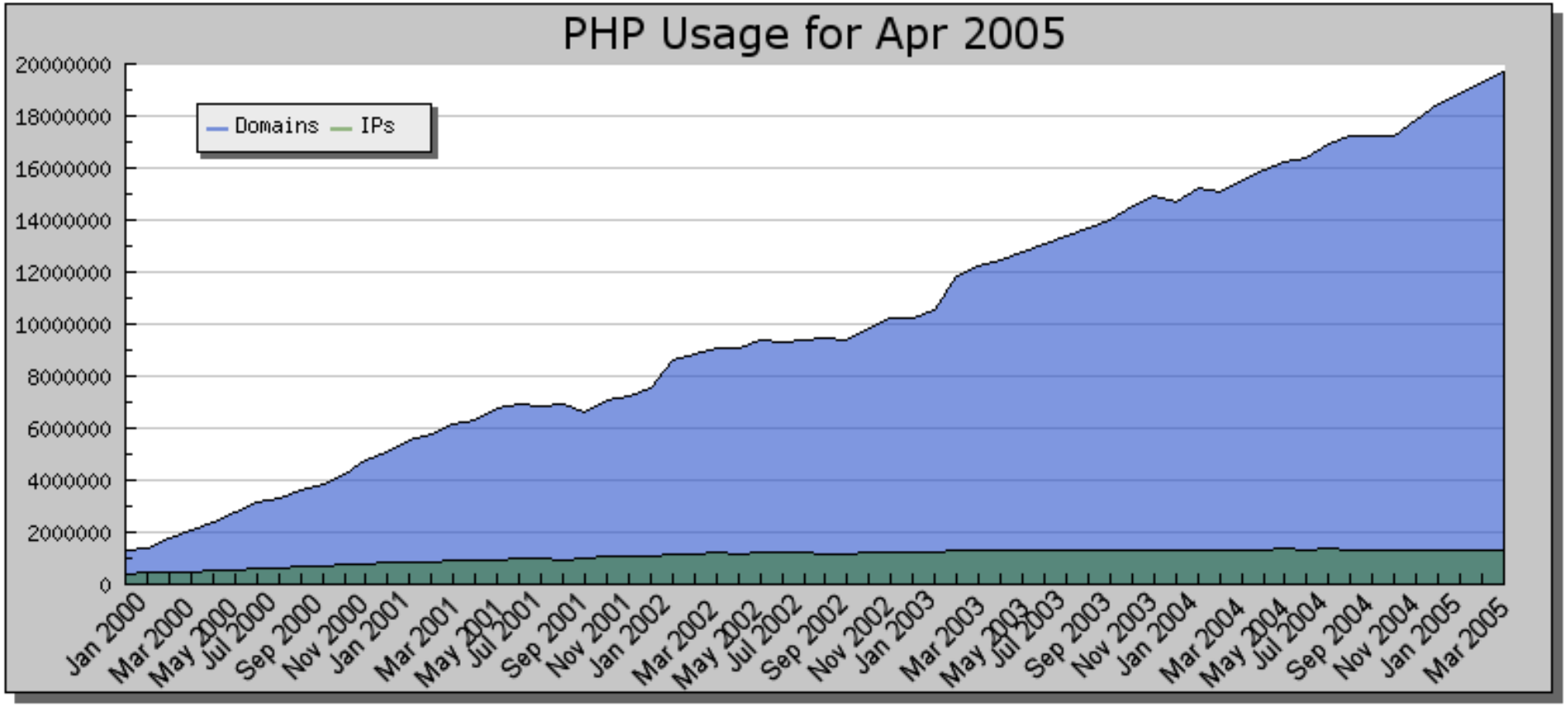
- **Server-side scripting**
  - To generate dynamic Web pages
  - Became popular also for generating XML documents, PDF files, GIF images, connect to other network services (like LDAP), and more
- **Command-line scripting**: can run scripts from the command line, much like Perl, or the Unix shell
- **Client-side GUI applications**, see PHP-GTK for details

Some of PHP's Strengths

- Performance
- Database integration
- Easy of learning and using PHP
- Portability
- Open source code

A Short History and the Growth of PHP

- Conceived in 1994 by Rasmus Lerdorf, a real zenith in 2002
- PHP originally stood for *Personal Home Page*, but changed following the GUI *PHP Hypertext Preprocessor*



Sample Application

```
<form action="processororder.php" method=post>
<table border=0>
<tr bgcolor=#cccccc>
  <td width=150>Item</td>
  <td width=15>Quantity</td></tr>
<tr>
  <td>Tires</td>
  <td align="center"><input type="text"
    name="tireqty" size="3" maxlength="3"></td></tr>
<tr>
  <td>Oil</td>
  <td align="center"><input type="text"
    name="oilqty" size="3" maxlength="3"></td></tr>
<tr>
  <td>Spark Plugs</td>
  <td align="center"><input type="text"
    name="sparkqty" size="3" maxlength="3"></td></tr>
<tr>
  <td colspan="2" align="center"><input
    type="submit" value="Submit Order"></td></tr>
</table></form>
```

Sample Application (cont.)

- The name of the PHP script that will process the order, *not* the URL where the user data will be sent

```
action="processorder.php"
```

- Keep in mind the names of the form fields for later call within a PHP code

Processing the Form: Embedding PHP in HTML

The `processorder.php` file:

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<?php
  echo '<p>Order processed.<p/>';
?>
</body>
</html>
```

Raw PHP is Unvisible at the Client Machine

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<p>Order processed.<p/>
</body>
</html>
```

PHP Basic Constructs

- Different tag styles:
  - XML style: `<?php ... ?>`
  - Short style: `<? ... ?>`
    (if `short_tabs` enables in config)
  - SCRIPT style:
    `<script language='php'> ... </script>`
  - ASP style: `<% ... %>`
    (if `asp_config` enabled)
- Blocks of code: use a semicolon at the end of each statement
- Whitespaces ignored - use them for readability only
- Comments: mulitline `/* .. */`, single line with `//..` or with `# ..`

Adding Dynamic Content

Provide dynamic content to a site's users:

- According to a user's need
- Over time

```
<?php
  echo '<p>Order processed at ';
  echo date('H:i, jS F');
  echo '</p>';
?>
</body>
</html>
```

Accessing Form Variables

Basically, access a form field using a PHP variable whose name *relates* to the name of the form field

- Variables in PHP start with a dollar sign `$`

Method 1: The same name preceeded with `$`, like `$tireqty`

- The form variables are all passed into your script (like arguments are to functions)
- Convenient, but error-prone: could be easily mixed-up with user defined global variables
  - To avoid it, initialize your own variables in time

Method 2: The name of a variable as a member identifier of array

- Form variables are stored in one of the arrays `_GET`, `_POST`, or `_REQUEST`, depending on the transfer method
  - `$_POST['tireqty']`, or `$HTTP_POST_VARS['tireqty']`

Accessing Form Variables: The Running Example

```
<?php
  //create short variable names
  $tireqty = $HTTP_POST_VARS['tireqty'];
  $oilqty = $HTTP_POST_VARS['oilqty'];
  $sparkqty = $HTTP_POST_VARS['sparkqty'];

echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br/>';
echo $oilqty.' bottles of oil<br/>';
echo $sparkqty.' spark plugs<br/>';
?>
```

- '.' is a string concatenation operator

Variable Types

PHP is a very weakly typed language

- No need to declare a variable before using it
- The type of a variable is determined by the value assigned to it (on-the-fly change of type)
  `$totalqty = 0;` ⇒ of type integer
  `$totalamount = 0.0;` ⇒ of type double
  `$totalqty = 'Hi';` ⇒ turned into a string

Built-in data types:

- Integer, Double, String, Boolean, Array, Object, NULL, etc.

More on Variables

Type casting

- `$totalqty = 0;`
  `$totalamount = (double)$totalqty;`
  - The type of $totalqty remains integer

Variable variables

- Alow to change the name of variables dynamically
  - `$varname = 'totalqty';`
    then `$$varname = 5` same as if `$totalqty = 5`

Constants

Example:

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

- Call `phpinfo()` to retrieve information about built-in variables, constants and much more in PHP

Variable Scope

Refers to the places within a script where a given variable is visible:

- **Superglobals** are visible everywhere within a script, like are arrays:
  - $_GET, $_POST, $_REQUEST, $GLOBALS, $_SERVER, $_ENV, $_FILES, $_COOKIE, $_SESSION
- **Global variables** are not visible inside functions
- **Variables** used inside functions are **local to the function**

Operators

- Arithmetics (+, -, *, /, %)
- Concatenation: `$a = "Great "; $b = "thing"; $a.$b;`
- Assignment: `$b = 10 + ($a = 5);` // equal 15
  - Combination assignment: (+=, -=, *=, /=, %=, .=)
  - Pre- and post-increment and decrement: `$a=5;`
    ```
    echo ++$a;
    echo $a++;
    ```
  - References: `$a = 5; $b = &$a; $a = 6;`
    (both $a and $b will change to 6)

Operators (cont.)

- Comparison (==, ===, !=, <>, >, >=, <, <=)
  === compares for identical operands, i.e., equal and of same type
- Logical (!, &&, ||, and, or)
- Ternary ?:, say `($grade > 5 ? 'passes' : 'failed');`
- Error suppression @: `$a = @(5/0);` // divide-by-zero!
- Execute commands inside ‘here comes a command line‘ (backticks): `$out = ‘ls -al‘; echo $out;`

Using Operators: Working Out the Form Totals

```
$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br/>';

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

echo 'Subtotal: $'.number_format($totalamount,3).'<br/>';

$taxrate = 0.10; // local sales tax is 10%
$totalamount = $totalamount * (1 + $taxrate);
echo 'Total tax: $'.number_format($totalamount,2).'<br/>';
```

Variable Functions

- `string gettype(mixed var);`
- `bool settype(mixed var, string type);`
- `is_array()`, `is_double()`, `is_string()`, etc.
- `bool isset(mixed var);`
- `boolean empty(mixed var);`

Flow-Control Structures

Conditionals

- if, else, elseif, switch

Iterations

- while loops
- for and foreach loops
- do..while loops

Stop Execution Statements

- `break` to leave the loop and continue at the next line after the loop
- `continue` to jump to the next loop iteration
- `exit` to break out of the entire PHP script

Any Questions?

- Next week about PHP and its communication to MySQL web databases