PHP Basics

- We will look at the language more formally later.
- For now
  - become familiar with the programming model
  - get familiar with server-side programming
  - get used to handling form submission

PHP: Hypertext Preprocessor

Some References:

`www.php.net`

`www.w3schools.com`

`http://www.oreillynet.com/pub/ct/29`

A PHP Program

- contained in an HTML document.
- All code is found inside:

  `<?php ... ?>` tags.
- conditional HTML
  - you can have PHP control what HTML actually makes it to the *output document*.

```
<?php if (foo)  { ?>
    <h3> foo is true!</h3>
<?php else ?>
    <h3> foo is false!</h3>
```

Web Server and PHP

- A client sends a request to a server, perhaps:

```
GET /myprog.php HTTP/1.1
```

- The server must be configured to recognize that the request should be handled by PHP.
- PHP reads the document `myprog.php` and produces some output (which is typically HTML).
  - all normal HTML tags are output without modification.

Our first PHP program?

```
<html>
<head>
<title>I am a PHP program</title>
</head>
<body>
<h3>PHP can handle HTML</h3>
</body>
</html>
```

A better example – try this

```
<html>
<head>
<title>I am a PHP program</title>
</head>
<body>
<?php phpinfo(); ?>
</body>
</html>
```

Server-side programming

- This is not like JavaScript
  - the browser does not understand PHP.
- You have to use a web server that supports php.
  - php *preprocesses* a file that contains HTML and code, and generates pure HTML (could actually be anything that the browser can understand).
  - If you “view source” in the browser, you can't tell the page was produced by php.

The Language in one slide

- Similar syntax to C/C++/Javascript
  - statements end with ';'
  - if, while, for, ... all just like we are used to.
  - assignment statements are the same.
- Variables are untyped (like Javascript)
  - can create a new variable by assigning a value.
  - variable names start with '$'
- arrays are different (associative arrays), but easy to get used to.

Generating HTML

- You can use functions **`echo`** , **`print`** and/or **`printf`** to generate HTML:

```
<?php
echo("<h3>Dynamic H3 tag</h3>");
print("<p>a paragraph produced with the print
  function</p>");
printf("<p>printf %s, works as well</p>\n",
       "(from C)");
?>
```

Jump right in!

```
<h3>Here are some lines</h3>
<?php
echo "<p>";
for ($i=0;$i<10;$i++) {
  echo "line " . $i . "<br>";
}
echo "</p>";
?>
```

variable names start with $
variables don't need to be declared.

string concatenation operator!

First Exercise

- Create a PHP script that produces an HTML table with the first 10 powers of two.
- You need to use the pow() function:
  - pow(x,y) = x^y

```
i          2^i
----------------------------------

1          2
2          4
3          8
4          16
5          32
6          64
7          128
8          256
9          512
10         1024
```

Non-obvious solution?

```
<table border=1>
<tr>
  <th>i</th>
  <th>2<sup>i</sup></th>
</tr>
<?php for ($i=1;$i<=10;$i++) { ?>
<tr>
  <td> <?php echo($i); ?></td>
  <td> <?php echo pow(2,$i);?></td>
</tr>
<?php } ?>
</table>
```

PHP style

- some PHP programmers think:
  - "I have an HTML document and I can add little bits of PHP code in special places"
- some others think:
  - "I have a PHP document in which I can add a few HTML tags".
- Use whatever style seems most comfortable to you.

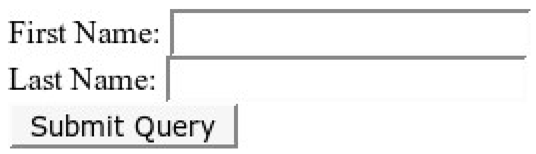
PHP and forms

- Recall that an HTML form submission looks something like this (GET method):

```
GET /somefile?x=32&name=Joe+Smith
```

- Typically a PHP script wants to get at the values the user typed into the form.
  - often the form itself came from a php script.

Form example



```
<form method="GET" action="form.php">
<p>
First Name: <input type="text" name="first"><br>
Last Name: <input type="text" name="last"><br>
<input type="submit">
</p>
</form>
```

Receiving the submission

- You can put the form in a file named "form.php", and set the action to also be "form.php".
  - The php file will receive the submission itself
  - You could also leave off the action attribute
- Unless we add some php code, all that will happen when the form is submitted is that we get a new copy of the same form.

Form Fields and PHP

- PHP takes care of extracting the individual form field names and values from the query.
  - also does urldecoding for us.
- A global variable named **`$_REQUEST`** holds all the form field names and values.
  - this variable is an associative array – the keys (indicies) are the form field names.

Getting the values

- To get the value the user submitted for the field named "first":

```
$_REQUEST['first']
```

- To get the value the user submitted for the field named "last":

```
$_REQUEST['last']
```

Adding some PHP to the form

- We could simply print out the values entered (as HTML):

```
<?php
 echo "<p>First name is ";
 echo  $_REQUEST['first'] . "</p>";


 echo "<p>Last name is ";
 echo $_REQUEST['last'] . "</p>";
?>
```

Or do it like this

```
<p>First name is <?php echo $_REQUEST['first'] ?>
</p>

<p>Last name is <?php echo $_REQUEST['last'] ?>
</p>
```

Make a php form handler

```
<form method="GET" action="form.php">
<p>First Name: <input type="text" name="first"><br>
   Last Name: <input type="text" name="last"><br>
   <input type=submit>
</p></form>
<?php
 echo "<p>First name is ";
 echo  $_REQUEST['first'] . "</p>";

 echo "<p>Last name is ";
 echo $_REQUEST['last'] . "</p>";
?>
```

Looking for Joe Smith

- We can easily turn this into a primitive login system.
  - we only allow Joe Smith to login
  - If the name is not Joe Smith, we send back the form along with a rude message.
- A real login system would not have the valid login names (passwords) *hard-coded* in the program
  - probably coming from a database.

Login handling form

```
<?php
 if (($_REQUEST['first'] == "joe") &&
     ($_REQUEST['last'] == "smith")) {
    echo "<p>Welcome back joe</p>;
 } else {
    ?>
   <p>You are not the correct person.</p>
   <p>Try again</p>
   <form method="GET" action="form.php">
   <p>First Name: <input type="text" name="first"><br>
   Last Name: <input type="text" name="last"><br>
   <input type=submit>
   </p></form>
<?php } ?>
```

Exercise

- Create a php script with a form where the user enters a number between 1 and 10.
- If they guess correctly, tell them!
- If they guess wrong – send the form back.

- Play with your php program directly (skipping the form) by constructing URLs manually.