

# An Introduction to PHP

---

Mary ET Boyle, Ph.D.

Department of Cognitive Science  
UCSD

# PHP is...

---

- ❑ A programming scripting language
  - ❑ It is used to extend XHTML
  - ❑ Used to create dynamic Web applications
-

- 
- ❑ Rasmus Lerdof created it in 1994
  - ❑ *Personal Home Page*
  - ❑ Apache Software Foundation project
  - ❑ Open source software (OSS)
  
  - ❑ PHP Hypertext Preprocessor
-

# Why learn it?

---

- Easy to use
  - Open source  
<http://www.apache.org>
  - Multiple platforms
  - Language support for databases  
(MySQL, Informix, Oracle, and Sybase)
-

## Internet-connected web browser

User enters Web address to PHP file

icogsci1.ucsd.edu/~cg3xzz/funstuff.php

Receive and interpret HTML file

## Web server

Receive request,  
find file, read it.

Execute PHP  
statements

Send results back

# Not all web servers

---

- Run software with built-in PHP
- Check with your ISP before your sign a contract.
- You can install it on your machine (not for novices to do this)
- Use version PHP 4.0 or greater

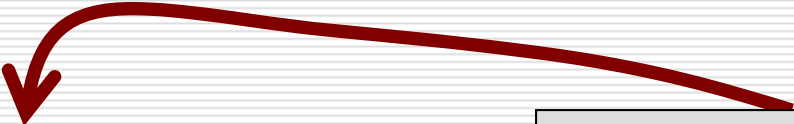
`phpinfo()`

---



first.php

---



Don't forget the extension.

```
<?PHP
    print ("A simple initial script");
?>
```

```
<script language="PHP">
    print ("A simple initial script");
</script>
```

---

# PHP debugging hints

---

- ❑ Pairs—  
quotation marks, parenthesis, brackets, etc. all need to be paired.
  - ❑ Semi-colons—  
the statements end with it;
  - ❑ Case sensitivity—  
PHP statements are case insensitive, but you need to adopt consistent case usage.
-

# PHP ignores blanks

---

```
print  
  ("a simple example");
```



These statements are the same.

```
print ("a simple example");
```

---

# Which version are you using?

---

```
<?php phpinfo(); ?>
```



Use this script to find out the version of PHP that is on the server.

---

# Sending Data to the Web Browser

---

□ PHP has two built in language constructs:

■ `echo ;`

■ `print ;`

```
echo 'hello, world!';
```

```
print "It's nice to see you.";
```



notice: either single or double quotation marks will work

---

# Embedding PHP statements in XHTML

---

```
<html> <head>
<title> Doing PHP stuff </title>
</head>
<body> <h1> Generating XHTML </h1>
<?php
    print("Using PHP is <i>great</i>");
    print("<ul>    <li>Speed</li>
           <li>Ease of use </li>
           <li>Functionality </li></ul>");
    print("</body></html>");
?>
```

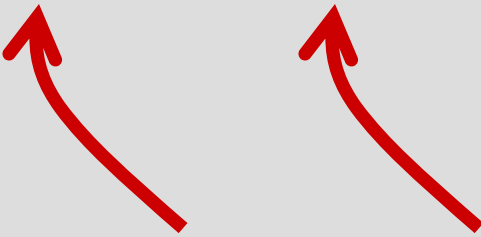
---

# Looking for an Escape!

## Using the \

---

```
print ( "<font color=\\"blue\\">" );
```



Sometimes you want to output an XHTML tag that also requires double quotation marks. You can use the backslash character to signal that the double quotation marks themselves should be output.

---

# PHP comments

---

```
<?php
// This is a comment
# This is a comment
/* This is block style of comments
   blah blah
*/
?>
```

```
<?php
    print ("hello"); // This is a comment
    echo `This is my PHP script!`;
?>
```

---

# Understanding PHP, XHTML, and White Space

---

- ❑ PHP – you can send data (a combination of XHTML tags and text) to the browser.
  - ❑ The browser will take that data and display it to the end user.
  - ❑ Therefore, you are using PHP to create the XHTML source of a web page.
-

# new line character (\n)

---

```
<?php
echo `This echo() statement runs
over the course of two lines!`;

echo "<br /> This is on one line.\n\n";

echo `Now I\'m done.`;
?>
```

for the new line to work, you need  
the **double** quotes!!

# Variables – what are they?

---

- ❑ widgets to temporarily store values
- ❑ values can be: numbers, text, etc.
- ❑ there are 8 types of PHP variables

- Boolean
- integer
- decimal (floating point)
- strings
- arrays
- objects
- resources
- NULL

← these are scalar variables

← these are multi-valued (non-scalar)

← special variable that has **no value**.

## syntactic variable rules:

---

- ❑ variable name must start with: **\$**
  - ❑ name can contain: strings, numbers, and underscore. i.e. **\$my\_report1**
  - ❑ the first character after the dollar sign **cannot** be a number
  - ❑ variables are case sensitive; **\$name** and **\$NAME** are different
  - ❑ variables can be assigned with **=**
-


# Working with `strings`

---

- ❑ a string is a quoted chunk of letters, numbers, and spaces.
  - ❑ These are all strings:
    - `'Mary'`
    - `'In a million years!'`
    - `'1,000,000'`
    - `'February 26, 2008'`
  - ❑ To make a string variable:
    - `$first_name = 'Mary';`
    - `$today = 'February 26, 2008';`
-

# Concatenating Strings

---

- An important tool when creating dynamic Web sites – it is like addition for strings.
- It is performed using the 

```
$first_name = 'Mary';  
$last_name = 'Boyle';  
$name = $first_name . $last_name;  
$name_2 = $first_name . ' ' . $last_name;
```

# Many functions for string manipulation in PHP

---

- Operators to determine the length of strings.
  - Get subsets of strings
  - Remove leading space characters
  
  - This is handy when you want to validate data that was entered in a xhtml form.
-

# Let's go over the assignment...

---

- ❑ The form – the action points to the php file.
  - ❑ Input:
    - Text boxes
    - Text areas
    - Radio buttons
    - Check boxes
    - Selection Lists
-

# Receiving Form Input into PHP Scripts

---

- ❑ To receive input from XHTML forms into your PHP script, use a PHP variable name that matches the variable defined in the form element's name argument.

```
<input type='radio' name='contact' value='yes'>
```

Then a form handling PHP script can access by using a variable called `$contact`

---