Intro to PHP

What Is PHP

- PHP
  - personal home page -> php: hypertext preprocessor
  - 1994 rasmus lerdorf
- server-side scripting language
  - interpreted
  - weekly-typed (values have types, not variables)
  - strong in text-processing (+ regular expressions)
- www.php.net/
  - release 5.3.2

Why PHP?

- widely popular
  - available at most web-hosting services
  - most popular server-side scripting language
- C heritage
  - like java, javascript, perl
- open source
  = free

How Does It Work?

- a web page contains PHP and XHTML
  - intertwined
1. client (=browser) requests a page
2. PHP scripts are executed on the server
3. they create XHTML markup
4. resulting page is send to the client
5. browser displays the page
- result = dynamic web site
  - content depends on the PHP script
  - e.g., may vary depending on a database query

Hello PHP World

- server-side page:
  <doctype >
  <html ...
  <title>ics php hi</title>
  <head>
  <php
  echo "hi";php hi</br>
  <p>hi from hi</p>
  </body></html>
- sent to client as:
  <doctype >
  <html ...
  <title>ics php hi</title>
  <head>
  <php hi</br>
  <p>hi from hi</p>
  </body></html>

Syntax

- <?php
  //some php script
  ?>
  - or (discouraged):
    - instead of "<?php
- can be anywhere on the page
  - typically functions are in page's <head>
- PHP scripts can be in separate files
  - e.g., for object classes
  - then, no "<?php ----- ?>
How To Code PHP?

• I recommend (on PC)
  – Eclipse with PHP
    • www.eclipse.org/pdt/
  – runs out of the box
    • no need to worry about server, etc.
  – but must learn Eclipse
    • but that’s good anyway
• on Mac, Unix?
  – any text editor
  – web server needed
• at UH
  – special web site needed
  – restricted, lots of paperwork

Basics

• echo string
  – or
• print string
  – insert string into XHTML markup
• comments
  – line comments start with // or #
    • # copyright 2010 ICS
  – /* display new line
  – multiline comments are enclosed in /* and */, e.g.
    • /* ICS
    = Insider-Created Scripts */

Comments

• # comments
  – customary in headings
  – copyright notices, etc.
  – don’t use them within code they distract
• // comments
  – used throughout code
  – comment what comes next on separate lines
  – leave lines empty to structure code is laziness
• /* ... */ comments
  – typically used for debugging purposes
    • to comment out code temporarily
    • to comment out debugging code in final product
• # copyright 2010 ICS

Strings

• delimited by " or '
  – " and ’ are similar
  – but different with variables – see later
• can stretch over several lines, e.g.,
  – " line 1
  – line 2"
• characters can be escaped with \ 
  – e.g., ‘I didn’t PHP’
  – e.g., “Say “Now I do” now!”
  – also, \n, \t, \r etc. may be needed
• special characters
  – \n new line, \t tab, \r carriage return...
  – while PHP does insert white space, XHTML typically ignores it

String Operations

• Concatenation with .
  – e.g., “ICS” . “is” . “fun”
• Functions
  – strlen(string)
  – empty(string)
  – substr(string, from, length)
  – strpos(string, pattern, from)
  – returns index of first pattern in string starting at optional from
  – explode(delimiter, string)
  – returns array (without delimiter)
  – implode(delimiter, array)
  – returns string with array elements concatenated with delimiter

Numbers

• Syntax
  – as usual: 3.14, -0.5e-5, ...
• Operators
  – +, -, *, / as usual
  – % modulo, ++1, --1
• Functions
  – round(number)
  – round(number, decimals)
  – number_format(number, decimals)
Variables

- **Syntax**
  - must start with $, e.g., $foo
  - followed by a letter or _
  - then any letters, digits or _
  - case-sensitive
  - use "camel notation"

- **Variables don't need to be declared**
  - just assign them using =

- **Variables don't have types**
  - the values they were assigned have types

- **Variable in a "..." string will be replaced by its value**
  - $name = "Hi!";
  - echo "I said $name"; //= 'I said "Hi!"'

- **Watch out: no replacement in a '...' string!**
  - $hi = '"Hi!"';
  - echo 'I said $hi'; //= 'I said $hi'

- **Variables can be references by PHP**
  - var op exp:
    - is the same as: var = var op exp.

Statements

- **Statements end with ;**
  - except when they use the { ... } form

- **Assignment**
  - uses =
  - not for equality ==

- **Operator-assignment is convenient**
  - e.g., $index += 2; // is $index = $index + 2
  - e.g., $name .= ","; // is $name = $name . ","

- **In general:**
  - var op exp:
    - is the same as:
      - var = var op exp.

Constants

- **Syntax**
  - first character letter or _
  - then any letters, digits or _
  - no $

- **Defined with**
  - define(name, value);

- **To print a constant, concatenate**
  - echo "pi=" . PI; // "pi=PI" won't work

- **I personally find all caps distracting**
  - what's so special about a constant to make it stand out?

HTML Forms

- **Allow user interaction**
  - java script/javascript.php: evaluate script

- **Contain widgets**
  - text fields
    - <input type="text" name="...">
  - pull-down menus
    - <select name="...">
    - <option value="...">choice-1</option>
    - <option value="...">choice-2</option>
  - radio buttons
    - <input type="radio" name="..." value="...">

- **Check boxes**
  - <input type="checkbox" name="..." value="...">

- **Submit button**
  - <input type="submit" name="..." value="Submit">

- (reset button – don’t use!) name can be referenced by PHP

- **Value** initial value, label

- **PHP page that handles a form accesses widgets’ value directly**

- **Special “superglobal” variable $_REQUEST**
  - associative array, i.e., array with string indices
  - indices are the name attributes of the widgets
  - e.g.,
    - the text in the field <input type="text" name="age">
  
- watch out: indices are case-sensitive

Forms Example

- **When user fills the form in this page and clicks "Submit"**
  - <html><body>
  
  - PHP page that handles a form accesses widgets’ value directly
  - Special "superglobal" variable $_REQUEST
  - "associative array", i.e., array with string indices
  - indices are the name attributes of the widgets
  - e.g.,
    - the text in the field <input type="text" name="age">
  
- watch out: indices are case-sensitive

- **Page response.php can access the widgets’ values**
  - <html><body>
Branches

- if (condition1) {...
  ...
} else if (condition2) {...
  ...
} else {...
  ...
} – else if and else are optional
  - elsif is also supported
  - e.g.,
  
  if ($gender == "female") {
    echo "Dear Madame,"
  } else if ($gender == "male") {
    echo "Dear Sir,"
  } else {
    echo "You forgot to choose the gender.";
  }

- switch as in Java
  – ugly because of break, or continue;

Loops

- while (condition) {
  ...
} while condition is true
  - as in Java
  - initial-statement, iteration-statement, executed always after block
  - can also use break, or continue;

- for (init-statement; condition; iteration-statement) {
  ...
} as in Java
  - loop while condition is true
  - initial-statement, executed once before block
  - iteration-statement, executed always after block
  - can also use break, or continue;

- foreach loop
  – iterates through array elements, e.g.,
  
  foreach ($array as $element) {
    echo "$element,"
  }
  
- Arrays

- indexed
  - integer index starting with 0, e.g., $days[0]
  - associative
  
  - string index e.g., $months['may']
  
  - to print, use {} braces: echo "1-st of {$months['may']}";
  
  - creating arrays
  
  - explicitly using array()
    - e.g., $days = array('mo', 'tu', 'we'); // $days[0] is 'mo', etc.
    - $days = array('jan'=>31, 'feb'=>28, 'mar'=>31); // $days['feb'] is 28

- with start index, e.g., $months = array(1=>'jan','feb','mar'); // $months[1] is 'jan'

- assigning subsequent elements
  - e.g., $days[$i] = 'mo';

- by assigning with indices
  - e.g., $days['jan'] = '31'; $days['feb'] = 28;

- with range()
  - $days = range(1, 7); // $days[0] is 1 ... $days[6] is 7

- Arrays (cont.)

- String is an array of characters
  - e.g., echo "last letter is [text[strlen($text) - 1]]"

- foreach loop
  - iterates through array elements, e.g.,
  
  foreach ($array as $index => $element) {
    echo "$index: $element;
  }

- Arrays can be sorted
  - sort($array)
    - sorts by value, does not sort the indices
  
  - asort($array)
    - sorts by value, maintains the indices
  
  - ksort($array)
    - sorts by indices
Multidimensional Arrays

- Multidimensional array is an array of arrays
- Subarrays can have different lengths
- Indexed and associative arrays can be mixed

  e.g.,
  - $usa = array('WI'=>'Wisconsin','IA'=>'Iowa','ID'=>'Idaho');
  - $canada = ('QC'=>'Quebec','YT'=>'Yukon');
  - $countries = array($usa,$canada);
  - foreach ($countries as $country => $states) {
      foreach ($states as $code => $state) {
          echo "$code stands for $state in $country"
      }
  }

Form Values As Arrays

- Some values of form widgets are accessed as an array
- Checkboxes with the same name attribute
  - <input type="checkbox" name="courses[]" value="215">
  - are accessed as multidimensional array
  - foreach ($_REQUEST['courses'] as $course) {
      echo "You choose ICS $course<br />
  }
  - only the checked boxes are in the array

- Drop-down menus with multiple selections
  - <select name="courses[]" multiple="multiple">
      <option value="215">ICS 215</option>
      <option value="315">ICS 315</option>
  </select>
  - are accessed the same way
  - only the selected menu items are in the array