PHP

Chapter 5

1 Server-Side Programming
2 PHP
- Arithmetic
- Variables
- Comments
- Boolean Logic
- Control Statements

3 Debug
4 Embedded PHP
5 Functions
6 Include
7 Arrays
8 Files
9 Objects

- HTML is a *markup language*.
- There are no for loops, function calls or if statements.
- HTML contents are **static**.
- Server-side languages allow you to create dynamic content.
- We will learn PHP in this course.
- Most php files are part php and part HTML. This mixture is called *embedded code*.

- PHP was developed by Rasmus Lerdorf in 1995 and originally meant "Personal Home Page."
- PHP now means "PHP Hypertext Preprocessor."
- The complete manual is located at http://www.php.net/manual/en/
- PHP is dynamically and weakly typed.
- PHP is an interpreted language.
- PHP is free, simple and widely available. Should be able to learn any other server-side language after learning PHP.

1. A client requests a particular page from a web server. Say it's index.php.
2. The web server executes the php file (php is an interpreted language).
3. The php file executes and produces content, usually HTML text. Programs that generate the content are called **server-side scripts**.
4. This HTML text is sent back to the client's machine. This content is called **dynamic content**.

Important Difference

- If you look directly at the php file on a web server, you would see php code.
- If you browse to a php page, your browser will show the HTML output.

- PHP code is saved in a file with a .php extension.
- Generally won't print out all HTML tags with php, e.g. <html> or </p>.
- Start off php code with <?php and end php code with ?>.
- The preprocessor finds all php code, executes it, and inserts the output into the HTML at that point.
- **print** - a function that outputs text, could also use *echo*. Escape characters are similar to Java: \" \n \' ...

```
<html><body><p> Bunch of normal HTML here.

<?php
print "Hello, world!";
?>

</p></body></html>
```

- PHP is **weakly** typed but every value has a type. All types can be found at http://www.php.net/manual/en/language.types.intro.php
- Manual **type casting** is possible but often unnecessary.
- Type casting has high precedence.
- Operator precedence is shown on http://www.php.net/manual/en/language.operators.precedence.php

```
<?php
print (int) 2.95;
print (int) "2.95";
print (float) "2.95";
print (int) "cs346";
print (int) 3 / 2;
print (int) (3 / 2);
print gettype(2.95);
print gettype("cs346");
print is_string(2.95);
print is_int(2.95);
?>
```

- PHP does arithmetic as you would expect it.
- Unlike Java, 3 / 2 is 1.5.
- http://www.php.net/manual/en/ref.math.php contains the built in math functions. No import or include statements are needed.

```
<?php
print 2.7 + "3.4";
print 7 / 2;
print rand(0, 10);
?>
```

- PHP has variables and they always start with a $ sign.
- A variable that has not been assigned a value will default at 0, 0.0, the empty string or empty array.
- Using an undeclared variable is a warning, not an error.
- Variable types are not explicitly declared.
- A variable can store any type and even change the type it stores throughout execution.
- PHP is stateless. Variables created on one page are not available on other pages (or on the same page after a reload).

```php
<?php
    $variable = 1 + 3;
    print $variable;
    $variable = 1.5 + 2.4;
    print $variable;
    $variable = 4 / "3";
    print $varaible;
    $variable .= $variable + 12; /* huh? */
    print $variable;
?>
```

- Be careful with + as it doesn't mean concatenation in PHP.
- If you want to concatenate strings, use the dot . operator. The dot operator has the same precedence as + and -.

```
<?php
    $x = "4";
    $y = "2";
    $z = $x + $y;
    print $z;
    $a = "hello";
    $b = ", how are you";
    $c = $a + $b;
    print $c;
?>
```

- References are also possible in PHP.
- Use the & sign to create a reference.

```
<?php
    $x = 12;
    $alias = &$x;
    $alias++;
    print $x;
?>
```

- Similar to Java, Strings are simply arrays of characters.
- Arrays are 0-indexed.
- strlen is a common function to get the length of the string.
- http://us3.php.net/strings contains the many String functions available to you.
- Some common ones you should know: strlen, strtoupper, str_replace, trim, strrev, ord, chr, strcmp, explode, implode, substr, strstr, htmlspecialchars.

```
<?php
    $str = "hello";
    print $str[1];
    print strlen($str);
    print $str[10];
    $x = "3 blind mice" + "5 golden rings";
    print $x;
?>
```

- An **interpreted string** is one where variables' names can be written inside of it. "hello, $user" is an example. 'hello, $user' is *NOT* interpreted.
- You can enclose the variable name with { }. "It is Bob's {$age}th birthday!"

```php
<?php
    $name = "bob";
    $printMe = "hello, $name";
    $alsoMe = 'hello, $name';
    print $printMe;
    print $alsoMe;
?>
```

- PHP has 3 types of comments.
- # starts a single line comment.
- // starts a single line comment.
- /* multi-line comment here */

```
<?php
    # comment
    // more comments
    /* lots
    of comments
    in this space!
    */
?>
```

- Boolean logic works mostly as you expect it.
- TRUE has a value of 1.
- FALSE has a value of 0.
- == and != ignore types.
- === and !== do not ignore types.
- FALSE values are 0-like or empty (array with no elements, undefined variables). All other values are TRUE.

- Control statements are almost identical to Java. Examples are shown on the next few slides.
- Be careful with equality checking.
- else if is legal but most use elseif.

```
<?php
    if(someBoolean){
     //code here
    } elseif(someOtherBoolean){
     //more code
    } else {
     //more code
    }
?>
```

- Control statements are almost identical to Java. Examples are shown on the next few slides.
- Be careful with equality checking.

```
<?php
    $x = 10;
    if($x = 20){
     print "equals";
    } else {
     print "not equals";
    }
?>
```

- Control statements are almost identical to Java. Examples are shown on the next few slides.
- Be careful with equality checking.

```
<?php
    for(initialization; bool; incrementStep){
     //code here
    }
    while(bool){
     //code here
    }
    do{
     //code here
    } while(bool);
?>
```

- An alternate way of using a control statement is with the : and end token.

```
<?php
    if(someBoolean){
     //code here
    } elseif(someOtherBoolean){
     //more code
    } else {
     //more code
    }
    /* the following chunk is identical code */
    if(someBoolean):
     //code here
    elseif(someOtherBoolean):
     //more code
    else:
     //mode code
    endif;
?>
```

- Syntax errors are displayed as part of the HTML output.
- A major error displays only the error message instead of any HTML.
- If you see *unexpected $end*, you probably forgot a ” or semi-colon or some other ending construct.

```
<?php
    /* add this to each file,
    it ensures all errors are printed */
    error_reporting(E_ALL | E_STRICT);
?>
```

- You should rarely print out HTML tags.
- You should embed PHP code around your HTML code.
- Whenever you put a starting element, e.g. (or {, you should always immediately place the ending element somewhere so it isn't forgotten.

```
<?php error_reporting(E_ALL | E_STRICT); ?>
<!DOCTYPE html><html>
<head><title>My page!</title></head>
<body>
<?php for($i = 0; $i < 10; $i++){ ?>
    <p>Printing number <?php print $i; ?>!!!</p>
<?php } ?>
</body>
</html>
```

- On the previous slide, there was a common statement: <?php print $i; ?>.
- Better to use an **expression block**. It injects a single PHP expression into the page.
- Do not forget the = in the expression block. <? /* php code */ ?> is identical to <?php /* php code */ ?>

```
<?php error_reporting(E_ALL | E_STRICT); ?>
<!DOCTYPE html><html>
<head><title>My page!</title></head>
<body>
<?php for($i = 0; $i < 10; $i++){ ?>
    <p><?= $i ?><sup>2</sup> is <?= $i*$i ?>!!!</p>
<?php } ?>
</body>
</html>
```

- Functions are allowed in PHP and they can have parameters and return a value.
- No types are declared in the parameter list.

```
<?php
    function posquadratic($a, $b, $c){
     return ((-1*$b) + sqrt($b*$b - 4*$a*$c)) / (2*$a);
    }
    function negquadratic($a, $b, $c){
     return ((-1*$b) - sqrt($b*$b - 4*$a*$c)) / (2*$a);
    }
    $mya = 10;
    $myb = 20;
    $myc = 25;
    $rootOne = posquadratic($mya, $myb, $myc);
    $rootTwo = negquadratic($mya, $myb, $myc);
?>
```

- Default parameter values are available for all trailing parameters.
- If no values are passed in, default values are used.

```
<?php
    function posquadratic($a, $b = 1, $c = 2){
     return ((-1*$b) + sqrt($b*$b - 4*$a*$c)) / (2*$a);
    }
    function negquadratic($a, $b = 1, $c = 2){
     return ((-1*$b) - sqrt($b*$b - 4*$a*$c)) / (2*$a);
    }
    $mya = 2;
    $myb = 3;
    $rootOne = posquadratic($mya);
    $rootTwo = negquadratic($mya, $myb);
?>
```

- Most arguments are passed **by value** by default.
- Changing an object passed into a function will change the original object.

```
<?php
    function byValue($a){
     $a *= 2; /* original variable is unchanged */
    }
    function byRef(&$a){
     $a *= 2; /* original variable is changed */
    }
    $mya = 2;
    byVal($mya);
    print $mya; //still 2
    byRef($mya);
    print $mya; //now 4
?>
```

- PHP has 2 scope levels, *global* and *local*.
- A variable “created” inside a for loop or if statement is accessible outside the for loop or if statement.

```
<?php
    function func($a){
        for($i=0; $i<$a; $i++){
            $count += $i;
        }
        return $count;
        // if warnings are suppressed, 45 is returned
        /* if warnings are not suppressed, 45
         is returned and a warning is displayed. */
    }
    $sum = func(10);
    print $sum;
?>
```

- Global variables are not assumed to be used in functions.
- Must explicitly state you are using a global variable.

```
<?php
    function func($a){
        return $a + $myGlobalVar;
    }
    $myGlobalVar = 20;
    $x = 10;
    print func($x);
?>
```

- Global variables are not assumed to be used in functions.
- Must explicitly state you are using a global variable.

```php
<?php
    function func($a){
     global $myGlobalVar;
     return $a + $myGlobalVar;
    }
    $myGlobalVar = 20;
    $x = 10;
    print func($x);
?>
```

- PHP allows you to include other files.
- This helps with redundant code or a file of special functions you've created.
- Useful for header and footer information or any other text that is displayed again and again.
- include_once is a useful function to ensure a file is only included once.
- require ensures a file is included else an error is displayed.

```php
<?php
    include("header.php");
    //bunch of php code here
    include("footer.php");
?>
```

- PHP has a built in *array* type.
- Arrays are indexed the same as Java. A few minor differences are shown in the examples below.

```
<?php
    $arr = array(); //empty array
    $arr = array(1, 2, 3, 4);
    print $arr[2];
    $arr[2] = 50;
    $arr[] = 90; //append 90 to array
    //count($arr) == 5
    for($i = 0; $i < 5; $i++){
     print $arr[$i];
    }
?>
```

- Arrays need not store elements in contiguous fashion.
- Many built in functions to manipulate and use arrays. Can find those at http://www.php.net/manual/en/ref.array.php
- Functions you should know: array_push, array_pop, array_unshift, array_shift, array_reverse, array_search, count, explode, implode, list, sort and rsort.
- unset($someArray[$someIndex]) makes a hole in the array.

```php
<?php
    $arr = array(1, 2, 3, 4);
    $arr[] = 50;
    $arr[10] = 100;
    $arr[] = 200;
    for($i = 0; $i < 20; $i++){
     print "$i : {$arr[$i]}" . "<br />";
    }
    print_r($arr); //can you write this function?
?>
```

- A foreach loop is built into PHP.
- Syntax is similar to the foreach loop in C#.
- The foreach loop cannot modify individual elements.

```php
<?php
    $arr = array(1, 2, 3, 4, 6, 8, 10);
    foreach($arr as $num){
     print $num . " ";
    }
?>
```

- We will eventually talk about databases but we'll start with file i/o.
- http://www.php.net/manual/en/ref.filesystem.php shows the many file functions.
- Will use *file_get_contents* often. It returns a long string.
- *file* returns an array of strings. Each line is an element in the array.
- One way to suppress error messages is to use @.
- We save files using *file_put_contents* passing in the filename followed by the text. Be careful, it will overwrite without warning.

```php
<?php
    $text = @file_get_contents("input.txt");
    file_put_contents("output.txt", $text);
?>
```

- **scandir** - accepts a directory name and returns an array of all files in that directory. The current and parent directory are always included in the array.
- **glob** - accepts a string with wildcards to match a set of files.

```
<?php
    $music = glob("*.mp3");
    foreach($music as $song){
     print $song; ?> <br /> <?
    }
?>
```

- PHP is an object-oriented programming language.
- Classes act similarly to Java classes with minor syntax differences.
- Methods and variables can be *public* or *private*.
- Objects are created using the **new** keyword.
- The *constructor* method looks like:
  public function __construct()
- Instance variables/methods are referenced with the $this variable. You can access a variable called $temp by using: $this->temp.
- **var_dump** will print out a detailed display of the state of an object.
- PHP features inheritance, static members, interfaces and abstract classes.

```
<?php
//assume this is in MyClasses.php
    class Rectangle{
        private $length;
        private $width;

        public function __construct(){
            $this->length = 0;
            $this->width = 0;
        }

        public function setLength($len){
            if($len > 0){
                $this->length = $len;
            }
        }
    }
?>
```

```php
<?php
//assume this is in MyClasses.php
    class Rectangle{
        //continued from previous slide
        public function getLength(){
            return $this->length;
        }

        public function getArea(){
            return $this->length * $this->width;
        }

        public function __toString(){
            //this prints out when you try and
            //print the object directly
            return "Rectangle: " . $this->length .
                   " by " . $this->width;
        }
    }
?>
```

```
<?php
    include("MyClasses.php");
    $myRect = new Rectangle();
    $myRect->setLength(20);
    $myRect->setWidth(10); //assume it was implemented
    print $myRect->getLength();
    print $myRect->getArea();
?>
```