

G22.3033-002 Scripting Languages

6/19/2008
Server-Side Scripting (PHP)

© Martin Hirzel G22.3033-002 NYU 6/19/2008 1

Outline

- HTML Basics
- PHP Basics
- Programming in the large

© Martin Hirzel G22.3033-002 NYU 6/19/2008 2

HTML

About HTML

- HyperText Markup Language
 - HyperText = contains clickable links
 - Markup Language = structuring and rendering instructions
- Used for most of the internet
 - Static HTML = simple file retrieved from server's disk, or generated by server-side script, e.g., PHP
 - Dynamic HTML = HTML containing client-side script

© Martin Hirzel G22.3033-002 NYU 6/19/2008 3

HTML

Related Languages

- Markup languages: troff, LaTeX, texinfo
- 1991, Tim Berners-Lee published description of HTML as application of SGML
- HTTP = HyperText Transfer Protocol
- Standards vs. Browser implementations
- 1996, XML = eXtensible Markup Language
 - Simpler variant of SGML
 - Used to define markup languages, such as HTML
- 2000, XHTML = HTML that conforms to XML

© Martin Hirzel G22.3033-002 NYU 6/19/2008 4

HTML

How to Write + Run Code

- See CIMS instructions: <http://www.cims.nyu.edu/systems/userservices/webhosting/>
- Create web site:


```
ssh access.cims.nyu.edu
ssh doowop1
mkdir $HOME/public_html $HOME/public_html/php
chmod 701 $HOME $HOME/public_html $HOME/public_html/php
cd $HOME/public_html/php
echo '<h1>Welcome to ' `pwd` '</h1>' > index.html
chmod 604 index.html
```
- Use a web browser to look at your web site:
 - http://www.cs.nyu.edu/~your_cims_id/php/index.html
 - http://www.cs.nyu.edu/~your_cims_id/php/

© Martin Hirzel G22.3033-002 NYU 6/19/2008 5

HTML

Structure of an HTML Document

© Martin Hirzel G22.3033-002 NYU 6/19/2008 6

HTML Lexical Peculiarities

Annotations in the screenshot:

- Comment: `<!-- ... -->`
- Element: `<table>...</table>`
- Start tag: `<table>`
- Attribute: `border=5`
- Contents (between matching tags): `<tr><th>English</th> <th>German</th></tr>`
- End tag: `</table>`
- Empty element: `<hr .../>`
- Entity: `&`

© Martin Hirzel G22.3033-002 NYU 6/19/2008 7

Reference HTML Documentation

- <http://www.w3schools.com/html/>

html	document	i	italic
head, title	header	a href	hyperlink
body	main part	ul	numbered list
h1, h2, h3	headings	ol	bullet list
p	paragraph	li	list entry
br	line break	table	table
hr	horizontal line	tr	table row
pre	preformatted	td	table data
b	bold	th	table head

© Martin Hirzel G22.3033-002 NYU 6/19/2008 8

HTML Input and Output

```

<html>
<head>
<title>Form test</title>
</head>
<body>
<form name="frmTest" action="script_url.php"
method="get"> <!-- or: method="post" -->
Text field: <input type="text" name="txt">
<br/>
Radio buttons:
A <input type="radio" name="rdo" value="a"/>
B <input type="radio" name="rdo" value="b"/>
<br/>
Check boxes:
C <input type="checkbox" name="chk" value="c"/>
D <input type="checkbox" name="chk" value="d"/>
<br/>
Submit button:
<input type="submit" value="submit"/>
</form>
</body>
</html>
    
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 9

HTML HyperText Transfer Protocol

```

graph TD
    User -- "enter form data" --> Browser
    Browser -- "click submit button" --> Server
    subgraph Server
        direction TB
        S[Run PHP, which creates HTML with result]
    end
    Server -- "HTTP response" --> Browser
    Browser -- "render markup" --> User
    
```

- HTTP GET: parameters encoded in URL
- HTTP POST: parameters in message header

© Martin Hirzel G22.3033-002 NYU 6/19/2008 10

Outline

- HTML Basics
- PHP Basics
- Programming in the large

© Martin Hirzel G22.3033-002 NYU 6/19/2008 11

PHP About PHP

- PHP: Hypertext Processor
 - Recursive acronym
 - HTML = HyperText Markup Language
- Claim to fame: simplicity
 - Script embedded in HTML generates HTML
 - Libraries for MySQL, Oracle, PDF, XML

```

graph TD
    User -- "click or type URL" --> Browser
    Browser -- "HTTP request" --> Server
    subgraph Server
        direction TB
        S[Run PHP to create HTML]
    end
    Server -- "serve HTML" --> Browser
    Browser -- "render markup" --> User
    
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 12

PHP

Related Languages

June 1995: PHP = Personal Home Page Tools, Rasmus Lerdorf, CGI scripts written in C

April 1996: PHP/FI scripting language, Rasmus Lerdorf

June 1998: Engine rewritten by Zeev Suraski and Andy Gutschans in Tel Aviv

May 2000: Parser rewritten again: "Zend Engine"

May 2004: PHP 5, current language version

- Evolution from script collection to scripting language
- Other server-side languages embedded in HTML: VisualBasic (ASP), Java (JSP)

© Martin Hirzel G22.3033-002 NYU 6/19/2008 13

PHP

Secure Your Website!

- See CIMS instructions: <http://www.cims.nyu.edu/systems/userservices/webhosting/>
- Put the following in `$HOME/public_html/php/.htaccess`:


```
AuthType Basic
AuthUserFile /home/your_cims_id/.htpasswd
AuthName "Members ONLY"
require valid-user
```
- Create a user name and password:


```
/usr/bin/htpasswd -c $HOME/.htpasswd user_name
# make up a new password for user_name, don't forget it
chmod 604 $HOME/.htpasswd $HOME/public_html/php/.htaccess
```
- Use a web browser, that will request authorization: http://www.cs.nyu.edu/~your_cims_id/php/index.html

© Martin Hirzel G22.3033-002 NYU 6/19/2008 14

PHP

How to Write + Run Code

- Put the following in `$HOME/public_html/php/hello.php`:


```
<html><body>
<?php
if(!empty($_GET['who'])) { echo "Hi, ".$_GET['who'].";" }
?>
<form action="<?php echo $_SERVER[PHP_SELF]; ?>" method=get>
Who shall be greeted: <input type="text" name="who" />
</form>
</body></html>
```
- Set the permissions:


```
chmod 604 $HOME/public_html/php/hello.php
```
- Use a web browser to look at your php script: http://www.cs.nyu.edu/~your_cims_id/php/hello.php

© Martin Hirzel G22.3033-002 NYU 6/19/2008 15

Concepts

Embedding Code in Web Pages

- Browser doesn't see PHP script, only result of running script on server
- PHP code gets replaced by its own output (printed by "echo" and other functions)
- Four styles of embedding PHP:
 - XML style (preferred!) `<?php ... ?>`
 - SGML style `<? ... ?>`
 - ASP style `<% ... %>`
 - Script style `<script language="php"> ... </script>`
- Variant of SGML style: `<?=
shorthand for <? echo ... ?>`

© Martin Hirzel G22.3033-002 NYU 6/19/2008 16

Concepts

Self-Processing Pages

```

graph TD
    User[User] -- "click or type URL" --> Browser[Browser]
    Browser -- "HTTP GET" --> Server[Server]
    Server -- "Run PHP, which creates HTML with form" --> Browser
    Browser -- "render markup" --> User
    User -- "enter form data, submit" --> Browser
    Browser -- "HTTP GET or POST same URL + data" --> Server
    Server -- "Run PHP, which creates HTML with result" --> Browser
    Browser -- "render markup" --> User
  
```

- Same PHP script, different HTML response

© Martin Hirzel G22.3033-002 NYU 6/19/2008 17

PHP

Input and Output

- Input: EGPCS superglobals
 - `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`
 - `$_REQUEST`: union of G,P,C
 - `$_FILES`: contains uploaded files
 - `$_SESSION`: persistent state across loads
- Output: printed text in HTML document
 - `echo`, `print()`, `printf()`
 - `var_dump()`, `print_r()`: print human-readable form for debugging; warning: problems with cycles
 - `phpinfo()`: prints lots of diagnostic information

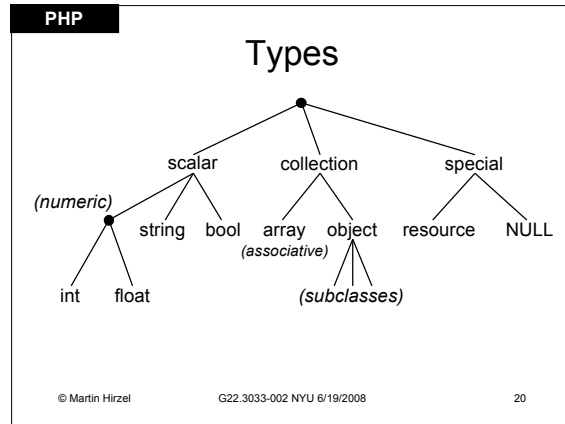
© Martin Hirzel G22.3033-002 NYU 6/19/2008 18

PHP

Lexical Peculiarities

- Embedded in HTML with `<?php ... ?>`
- Variables are case sensitive; classes, functions, and keywords are case insensitive
- All variables (including arrays) begin with dollar sign (\$), can be interpolated in string
- Semicolon required even after last statement in block, optional only before `?>`
- Single-line comments, or up to `?>`: `#, //`
- Multi-line comments, even across `?>`: `/*...*/`
- Literals: `"s", 's', multi-line string, true, null`
- Heredocs: continues after closing tag

© Martin Hirzel G22.3033-002 NYU 6/19/2008 19



PHP

Type Conversions

Value	bool	number	string
bool	false	Identity	""
	true	1	"1"
number	0	false	Represent value as string
	1	true	
	Other	true	Identity
string	""	false	0
	"0"	false	0
	Other	true	Numeric prefix
null		false	0
array	empty	false	Error
	Other	true	"Array"
object	empty	false	Error
	Other	true	"Object"

© Martin Hirzel G22.3033-002 NYU 6/19/2008 21

PHP

Variable Declarations

Implicit	<code>echo \$a + 1;</code> <code>\$b = 5;</code>	Read NULL if non-existent
Constant (global)	<code>define('PI', "3.14");</code> <code>echo PI</code>	Not a variable, don't need dollar sign (\$)
Global, used locally	<code>global \$g;</code> <code>\$g = \$g + 1;</code>	Otherwise, write creates new local \$g
Local, unlimited lifetime	<code>static \$s;</code> <code>\$s++;</code>	Otherwise, value forgotten after return

© Martin Hirzel G22.3033-002 NYU 6/19/2008 22

PHP

Operators

<code>new</code>	N	Create object
<code>[...]</code>	L	Array subscript
<code>++, --</code>	1	N Auto-increment / decrement
<code>~, @, (int), (array), ...</code>	1	N ~: bitwise negation; @: inhibit errors; (type): cast
<code>instanceof</code>	2	N Class/interface membership
<code>!</code>	1	R Logical negation
<code>*, /, %</code>	2	L Multiplicative
<code>+, -, .</code>	2	L Additive; .: string concatenation
<code><<, >></code>	2	L Bitwise shift
<code><, <=, >, >=</code>	2	N Comparison
<code>==, !=, <>, ===, !==</code>	2	N Identity; ==, !=: equal value + type
<code>&, ^, </code>	2	L Bitwise (not all same precedence)
<code>&&, </code>	2	L Logical (not all same precedence)
<code>?:</code>	3	L Conditional
<code>=, +=, -=, ...</code>	2	L Assignment
<code>and, or, xor</code>	2	L Logical (not all same precedence)
<code>,</code>	2	L List separator

© Martin Hirzel G22.3033-002 NYU 6/19/2008 23

PHP

Arrays

- Creation: `$a = array(1, 2, 3);`
`$b = array('cat'=>'meow', 'dog'=>'woof');`
- Indexing: e.g., `$a[2]`, `$b["dog"]`, `$b[cat]`
 - '3' and "3" and 3 are the same key
 - -3 is the same key as "-3"
 - Quotes around string keys are optional, and must be omitted for interpolation
 - Write to non-existent index inserts, e.g., `$a[3]=4;`
 - Write without index appends, e.g., `$a[]=4;`
- Remove: `unset($a[2])`, delete `unset($a)`
- Multiple assign: `list($x, $y) = array(1,2,3);`

© Martin Hirzel G22.3033-002 NYU 6/19/2008 24

PHP

Array Library Functions

<http://us.php.net/manual/en/ref.array.php>

<code>range(low, high, [, step])</code>	Create array
<code>count(var [, mode])</code>	Size
<code>array_keys(input [, ...])</code>	Indices
<code>array_values(input)</code>	Contents
<code>array_key_exists(key, search)</code>	Contains
<code>sort(array [, sort_flags])</code>	By values
<code>rsort(array [, sort_flags])</code>	Descending
<code>usort(array, cmp_function)</code>	By user function
<code>array_push(array, var [, ...])</code>	Add to end
<code>array_pop(array)</code>	Remove from end
<code>array_walk(array, func [, data])</code>	Mapping

© Martin Hirzel G22.3033-002 NYU 6/19/2008 25

PHP

Control Statements

Conditional	<code>if (expr) ... [elseif (expr) ...] [else ...]</code> <code>switch(expr) {case expr:... default: ...}</code>
Fixed-iteration loops	<code>for (expr; expr; expr) ...</code> <code>foreach (\$arr as \$val) ...</code> <code>foreach (\$arr as \$key => \$val) ...</code>
Indefinite loops	<code>while (expr) ...</code> <code>do ... while (expr);</code>
Unstructured control	<code>break [expr];</code> # <i>expr</i> = loop levels to skip <code>continue [expr];</code> # <i>expr</i> = loop levels to skip <code>exit [expr];</code> # <i>expr</i> = error message <code>return [expr];</code> # <i>expr</i> = return value
Directive	<code>declare (directive) ...</code> # rarely used

© Martin Hirzel G22.3033-002 NYU 6/19/2008 26

PHP

Alternative Control Syntax

<code>if (\$x < \$y) :</code>	<code>if (\$x < \$y) {</code>
<code>echo "then branch";</code>	<code>echo "then branch";</code>
<code>\$min = \$x;</code>	<code>\$min = \$x;</code>
<code>else:</code>	<code>} else {</code>
<code>echo "else branch";</code>	<code>echo "else branch";</code>
<code>\$min = \$y;</code>	<code>\$min = \$y;</code>
<code>endif;</code>	<code>}</code>

Also available for other control statements, e.g.:

- `while (expr) : ... endwhile;`
- `for (expr; expr; expr) : ... endfor;`
- `switch (expr) : case expr:... default:... endswitch;`

© Martin Hirzel G22.3033-002 NYU 6/19/2008 27

PHP

References

“Variable variable”	“Alias”
- Store name of one variable as string in other variable	- Make two variables refer to the same memory location
- Also known as “soft reference”	- Also known as “hard reference”

<code>\$x = 123;</code>	<code>\$x = 123;</code>
<code>\$r = 'x';</code>	<code>\$r = &\$x;</code>
<code>echo \$\$r;</code>	<code>echo \$r;</code>
<code>\$r = '';</code>	<code>unset \$r;</code>

© Martin Hirzel G22.3033-002 NYU 6/19/2008 28

PHP

Writing Subroutines

- Declaration: `function [&]id(arg*) { ... }`
 - To return a value: `return expr;`
 - `&`: return alias for result (hard reference)
- Arguments: `arg ::= [&]$id [= expr]`
 - Call-by-value, even for arrays
 - `&`: call-by-reference
 - `[= expr]`: optional parameter, default value
 - Empty (*arg**): `$my_array = func_get_args();`
- Variable functions: same as variable variables
- Creating new variable function from strings:


```
$x=create_function('$a','echo $a;'); $x('hi');
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 29

Outline

- HTML Basics
- PHP Basics
- Programming in the large

© Martin Hirzel G22.3033-002 NYU 6/19/2008 30

PHP

Finding PHP Mistakes

- If script encounters error on server, the browser just gets empty HTML ⇒ not helpful!
- For compile errors: run at command line
 - `php -f file` Parse and execute file
 - `php -s -f file` Syntax highlighted source
 - `php -l -f file` Lint (check syntax without running)
- For logic errors:
 - Use `echo` statements to see what gets executed
 - Use `var_dump()` calls to inspect data structures
- To view source in browser:


```
ln -s script.php script.phps
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 31

PHP

Structure of a PHP Application

- Literal inclusion of code from file:
 - `require 'fileName'`; fatal if non-existent
 - `include 'fileName'`; warn if non-existent
 - `require_once 'fileName'`; no effect if repeated
 - `include_once 'fileName'`; no effect if repeated
- Use `@include` to suppress the warning
- Convention: `fileName` extension `.inc`
- No separate scope / namespace for included code, may cause proliferation of globals

© Martin Hirzel G22.3033-002 NYU 6/19/2008 32

PHP 5

Using Objects

<code>require_once 'Apple.inc';</code>	Include file
<code>\$a1 = new Apple(150, "green");</code> <code>\$a2 = new Apple(150, "green");</code>	Constructor calls
<code>\$a2->color = "red";</code> <code>\$varvar = "weight";</code> <code>\$a2->\$varvar = 220;</code>	Property access
<code>echo \$a1->prepare("slice") . "
\n";</code> <code>echo \$a2->prepare("squeeze") . "
\n";</code>	Method calls

© Martin Hirzel G22.3033-002 NYU 6/19/2008 33

PHP 5

Defining Classes

```
class Fruit {
    var $weight = 0;
    function __construct($weight) {
        $this->weight = $weight;
    }
    function pluck() {
        return "fruit(" . $this->weight . "g)";
    }
    function prepare($how) {
        return $how . "d " . $this->pluck();
    }
}
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 34

PHP 5

Inheritance in PHP

```
class Fruit {
    var $weight = 0;
    function __construct($weight) {
        $this->weight = $weight;
    }
    function pluck() {
        return "fruit(" . $this->weight . "g)";
    }
    function prepare($how) {
        return $how . "d " . $this->pluck();
    }
}
class Apple extends Fruit {
    var $color = "green";
    function __construct($weight, $color) {
        $this->weight = $weight;
        $this->color = $color;
    }
    function pluck() {
        return $this->color . " apple";
    }
}
```

© Martin Hirzel G22.3033-002 NYU 6/19/2008 35

PHP 5

More on Classes

- Modifiers: method (`abstract`, `final`), property (`public`, `private`, `protected`, `const`), both (`static`)
- Static member access: `::`, `self::`, `parent::`
- Interface: like in Java, all methods implicitly abstract; class can extend class and implement interfaces
- Access to non-existent property turns into method call: `__get($propName)` or `__set($propName, $value)`
- Called on object death: `__destruct()`
- Introspection: `class_exists`, `get_declared_classes`, `get_class_methods`, `get_class_vars`, `get_parent_class`, `is_object`, `get_class`, `method_exists`, `get_object_vars`

© Martin Hirzel G22.3033-002 NYU 6/19/2008 36

PHP

Scopes and Visibility

- Locals: scope is entire function, not just block
- Globals:
 - “`global $x;`” is shorthand for
“`$x = &$GLOBALS['x'];`”
 - `register_globals` in `php.ini` causes EGPCS to be spilled into globals; that's bad for security
- Nested functions:
 - Inner function does not see outer locals/arguments
 - Inner function globally visible after first call to outer
- Modules: `require/include` don't affect scoping
- Classes: `public`, `private`, `protected` properties

© Martin Hirzel

G22.3033-002 NYU 6/19/2008

37

Reference

PHP Documentation

- Language and library reference:
<http://www.php.net/manual/en/>
- CIMS web scripting instructions:
<http://www.cims.nyu.edu/systems/userservices/webhosting/>
- Book: Programming PHP, 2nd edition [safari].
Rasmus Lerdorf, Kevin Tatroe, and Peter MacIntyre.
O'Reilly, 2006.
- Tutorial: <http://www.w3schools.com/>
- PEAR = PHP Extension+Application Repository:
<http://pear.php.net>

© Martin Hirzel

G22.3033-002 NYU 6/19/2008

38

Soap-box

Evaluating PHP

Strengths

- Simplicity
- Portability
- Large libraries
- Many database bindings
- Popularity

Weaknesses

- Error handling
- Lack of scalability
 - Compared to Java
- Low-level
 - Compared to Ruby on Rails or Google Web Toolkit

© Martin Hirzel

G22.3033-002 NYU 6/19/2008

39

Administrative

Last Slide

- Pick up graded hw01, hw02, quiz1
- Look at example solutions
- Course evaluations (you evaluate me)
- Today's lecture
 - HTML, HTTP
 - Server-side scripting
 - PHP
- Next lecture
 - Client-side scripting
 - JavaScript
- Next+1 lecture
 - Web applications
 - Databases

© Martin Hirzel

G22.3033-002 NYU 6/19/2008

40