

PHP

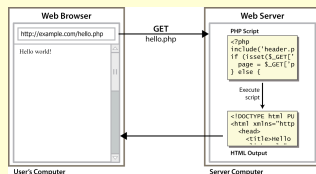
Based on content from *Web Programming Step by Step* by Marty Stepp, Jessica Miller and Victoria Kirst

PHP

- Originally: "Personal Home Page Tools"
- Now: "PHP Hypertext Preprocessor", a recursive acronym.
- A server-side scripting language
- Used to make web pages dynamic:
 - provide different content depending on context
 - interface with other services: database, e-mail, etc
 - authenticate users
 - process form information
- PHP code can be embedded in XHTML code

Request Life Cycle

- browser requests a .php file (dynamic content): server reads it, runs any script code inside it, then sends result across the network
 - script produces output that becomes the response sent back



Advantages

- Free and open source: anyone can run a PHP-enabled server free of charge
- Compatible: supported by most popular web servers
- Simple: lots of built-in functionality; familiar syntax
- Available: installed on most commercial web hosts

Hello World!

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

PHP

Hello, world!

output

- A block or file of PHP code begins with `<?php` and ends with `?>`
- PHP statements, function declarations, etc. appear between these endpoints
- Only works if page loaded from web server

Variables

```
$name = expression;
$user_name = "PinkHeartLuvr78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;
```

PHP

- names are case sensitive; separate multiple words with `_`
- names always begin with `$`, on both declaration and usage
- always implicitly declared by assignment (type is not written)
- a loosely typed language (like JavaScript or Python)

Types

- Basic types: int, float, boolean, string, array, object, NULL
 - Test what type a variable is with `is_type` functions, e.g. `is_string`
- PHP converts between types automatically in many cases:
 - string → int auto-conversion on +
 - int → float auto-conversion on /
- type-cast with `(type)`:
 - `$age = (int) "21";`

Operators

- + - * / % . ++ --
= += -= *= /= %= .=
== != === !== > < >= <=
&& || !
- `==` just checks value ("`5.0`" `==` `5` is TRUE)
- `===` also checks type ("`5`" `===` `5` is FALSE)
- many operators auto-convert types: `5 < "7"` is TRUE

Comments

- `#` *single-line comment*
- `//` *single-line comment*
- `/*` *multi-line comment* `*/`

String Type

- zero-based indexing using bracket notation

```
$favorite_food = "Ethiopian";  
print $favorite_food[2]; # h
```

- string concatenation operator is `.` (period)
 - `5 . "2 turtle doves" === "52 turtle doves"`
- strings inside `"` are interpreted
 - variables that appear inside them will have their values inserted into the string

```
$age = 16;  
print "You are ".$age." years old.\n";  
print "You are $age years old.\n"; # You are 16 years old.
```

- strings inside `'` are *not* interpreted:

```
print 'You are $age years old.\n'; # You are $age years old.\n
```

String Functions

```
$name = "Kenneth Kuan";  
$length = strlen($name); # 12  
$cmp = strcmp($name, "Jeff Prouty"); # > 0  
$index = strpos($name, "e"); # 1  
$first = substr($name, 8, 4); # "Kuan"  
$name = strtoupper($name); # "KENNETH KUAN"
```

Name	Java Equivalent
<code>explode, implode</code>	<code>split, join</code>
<code>strlen</code>	<code>length</code>
<code>strcmp</code>	<code>compareTo</code>
<code>strpos</code>	<code>indexOf</code>
<code>substr</code>	<code>substring</code>
<code>strtolower, strtoupper</code>	<code>toLowerCase, toUpperCase</code>
<code>trim</code>	<code>trim</code>

Boolean Type

- the following values are considered to be FALSE (all others are TRUE):
 - 0 and 0.0 (but NOT 0.00 or 0.000)
 - "", "0", and NULL (includes unset variables)
 - arrays with 0 elements
- can cast to boolean using `(bool)`
- FALSE prints as an empty string (no output); TRUE prints as a 1

```
$feels_like_summer = FALSE;  
$php_is_rad = TRUE;  
$student_count = 217;  
$nonzero = (bool) $student_count; # TRUE
```

Arrays

```
$name = array(); # create
$name = array(value0, value1, ..., valueN);

$name[index] # get element value
$name[index] = value; # set element value
$name[] = value; # append

$a = array(); # empty array (length 0)
$a[0] = 23; # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!"; # add string to end (at index 5)
```

Array Functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Associative Arrays

```
$name = array();
$name["key"] = value;
...
$name["key"] = value;

$name = array(key => value, ..., key => value);

$blackbook = array("marty" => "206-685-2181",
                  "stuart" => "206-685-9138",
                  "jenny" => "206-867-5309");

foreach ($blackbook as $key => $value) {
    print "$key's phone number is $value\n";
}

jenny's phone number is 206-867-5309
stuart's phone number is 206-685-9138
marty's phone number is 206-685-2181
```

if/elseif/else

```
if (condition) {
    statements;
} elseif (condition) {
    statements;
} else {
    statements;
}
```

Looping

```
for (initialization; condition; update) {
    statements;
}

while (condition) {
    statements;
}

do {
    statements;
} while (condition);

foreach ($array as $variableName) {
    ...
}

$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
    print "Moe slaps {$stooges[$i]}\n";
}

foreach ($stooges as $stooge) {
    print "Moe slaps $stooge\n"; # even himself!
}
```

Example

```
<?php
for ($i = 1; $i <= 3; $i++) {
    <h<?=$i ?>>This is a level <?=$i ?> heading.</h<?=$i ?>>
}
</body>
```

This is a level 1 heading.
This is a level 2 heading.
This is a level 3 heading.

Functions

```
function name(parameterName, ..., parameterName) {
    statements;
}

function quadratic($a, $b, $c) {
    return - $b + sqrt($b * $b - 4 * $a * $c) / (2 * $a);
}

$x = -2;
$a = 3;
$root = quadratic(1, $x, $a - 2);
```

Variable Scope

- variables declared in a function are local to that function
- variables not declared in a function are global
- if a function wants to use a global variable, it must have a global statement

```
$school = "UW"; # global
...
function downgrade() {
    global $school;
    $suffix = "Tacoma"; # local
    $school = "$school $suffix";
    print "$school\n";
}
```

CGI Super Global Arrays

- Associative arrays that contain information about HTTP request

Array	Description
\$_GET, \$_POST	parameters passed to GET and POST requests
\$_REQUEST	parameters passed to any type of request
\$_SERVER, \$_ENV	information about the web server
\$_FILES	files uploaded with the web request
\$_SESSION, \$_COOKIE	"cookies" used to identify the user (seen later)

Apache PHP

- Disabled by default
- http.conf
 - Uncomment
LoadModule php5_module libexec/apache2/libphp5.so