

CSE 154



LECTURE 14: INTRO TO PHP

URLs and web servers

```
http://server/path/file
```

- usually when you type a URL in your browser:
 - your computer looks up the server's IP address using DNS
 - your browser connects to that IP address and requests the given file
 - the web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you
- some URLs actually specify *programs* that the web server should run, and then send their output back to you as the result:
`https://webster.cs.washington.edu/cse190m/quote.php`
 - the above URL tells the server `webster.cs.washington.edu` to run the program `quote2.php` and send back its output

Server-Side web programming



- server-side pages are programs written using one of many web programming languages/frameworks
 - examples: [PHP](#), [Java/JSP](#), [Ruby on Rails](#), [ASP.NET](#), [Python](#), [Perl](#)
- the web server contains software that allows it to run those programs and send back their output
- each language/framework has its pros and cons
 - we will use PHP for server-side programming

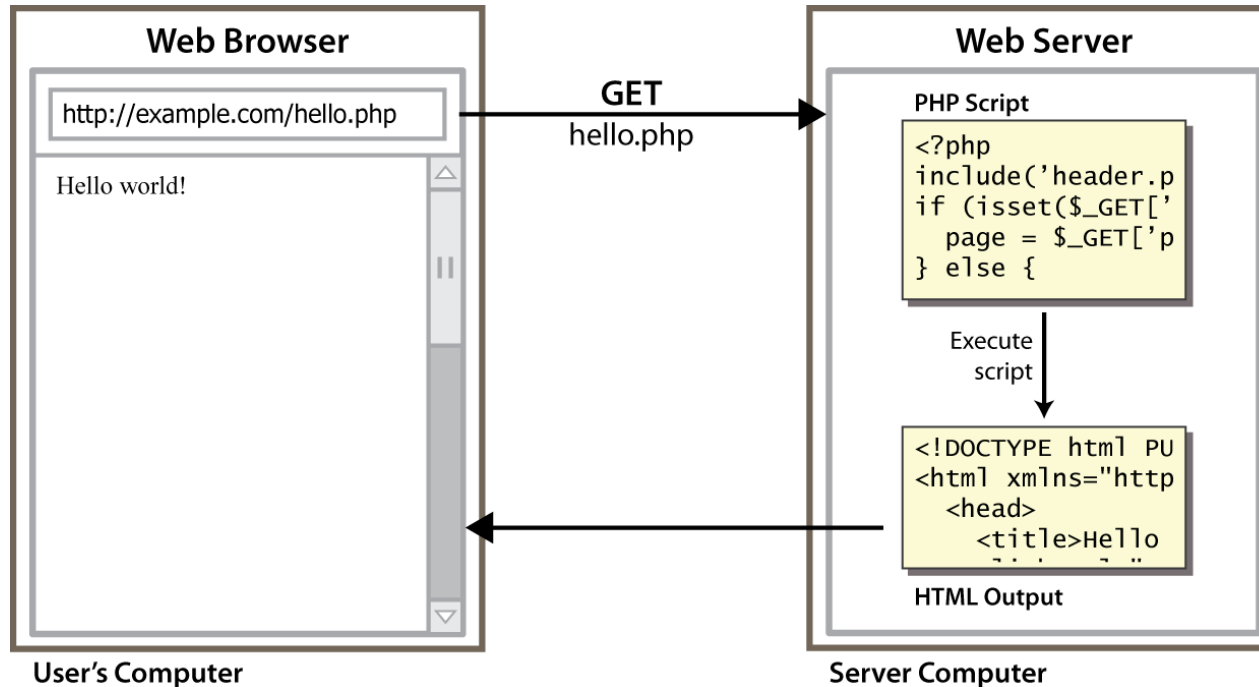
Why PHP?

There are many other options for server-side languages: Ruby on Rails, JSP, ASP.NET, etc.

Why choose PHP?

- 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 UW's servers (Dante, Webster) and most commercial web hosts
- **well-documented**: type `php.net / functionName` in browser Address bar to get docs for any function

Lifecycle of a PHP web request



- browser requests a `.html` file (**static content**): server just sends that file
- browser requests a `.php` file (**dynamic content**): server reads it, runs any script code inside it, then

Console output: print

```
print "text";
```

PHP

```
print "Hello, World!\n";
```

```
print "Escape \"chars\" are the SAME as in Java!\n";
```

```
print "You can have  
line breaks in a string.";
```

```
print 'A string can use "single-quotes". It\'s cool!';
```

PHP

Hello, World! Escape "chars" are the SAME as in Java! You can have line breaks in a string. A string can use "single-quotes". It's cool!

output

- some PHP programmers use the equivalent `echo` instead of `print`

Arithmetic Operations

- `+ - * / %`
`. ++ --`
`= += -= *= /= %= .=`
- many operators auto-convert types: `5 + "7"` is `12`

Variables

```
$name = expression;
```

PHP

```
$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
- implicitly declared by assignment (type is not written; a "loosely typed" language)

Types

- basic types: int, float, boolean, string, array, object, NULL
 - test what type a variable is with `is_type` functions, e.g. is string
 - gettype function returns a variable's type as a string (not often needed)
- PHP converts between types automatically in many cases:
 - `string` → `int` auto-conversion on + (`"1" + 1 == 2`)
 - `int` → `float` auto-conversion on / (`3 / 2 == 1.5`)
- type-cast with (*type*):
 - `$age = (int) "21";`

String type

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

```
# h
```

```
PHP
```

- zero-based indexing using bracket notation
- string concatenation operator is . (period), not +
 - `5 + "2 turtle doves"` produces `7`
 - `5 . "2 turtle doves"` produces `"52 turtle doves"`
- can be specified with `""` or `' '`

String functions

```
# index 0123456789012345
$name = "Austin Weale";
$length = strlen($name);           # 16
$cmp = strcmp($name, "Linda Guo"); # > 0
$index = strpos($name, "s");       # 2
$first = substr($name, 7, 4);      # "Weal"
$name = strtoupper($name);        # "AUSTIN WEALE"      PHP
```

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

Interpreted strings

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

- strings inside " " are interpreted
 - variables that appear inside them will have their values inserted into the string
- strings inside ' ' are not interpreted:

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

- if necessary to avoid ambiguity, can enclose variable in {}:

```
print "Today is your $ageth birthday.\n";      # $ageth not found  
print "Today is your { $age }th birthday.\n";      PHP
```

bool (Boolean) type

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

- the following values are considered to be FALSE (all others are TRUE):
 - 0 and 0.0
 - "", "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

for loop

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

PHP

```
for ($i = 0; $i < 10; $i++) {  
    print "$i squared is " . $i * $i . ".\n";  
}
```

PHP

if/else statement

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

PHP

- can also say `elseif` instead of `else if`

while loop (same as Java)

```
while (condition) {  
    statements;  
}
```

PHP

```
do {  
    statements;  
} while (condition);
```

PHP

- break and continue keywords also behave as in Java

Comments

```
# single-line comment

// single-line comment

/*
multi-line comment
*/
```

PHP

- like Java, but **#** is also allowed
 - a lot of PHP code uses **#** comments instead of **//**
 - we recommend **#** and will use it in our examples