Introduction to Web Programming

Lecture 6: Intro to PHP

6.1: Server-Side Basics

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:

  `http://www.polytech.unice.fr/~gaetano/tiei/iwp/php/login.php`

  - the above URL tells the server `www.polytech.unice.fr` to run the program `~gaetano/tiei/iwp/php/login.php` and send back its output

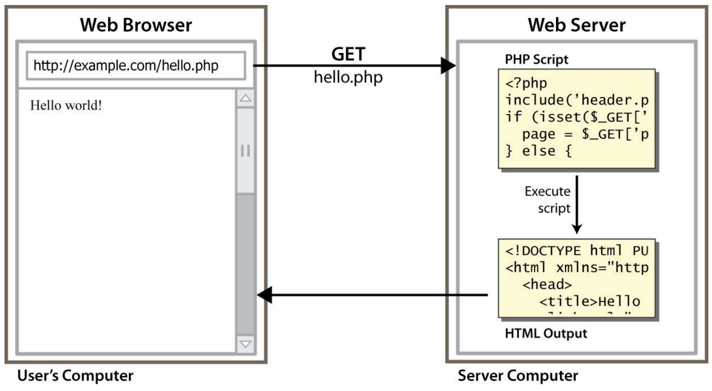
Server-Side web programming

- 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 use PHP for server-side programming in this course

What is PHP?

- **PHP** stands for "PHP Hypertext Preprocessor"
- 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 HTML code

Lifecycle of a PHP web request



```
<?php
include('header.p
if (isset($_GET['
  page = $_GET['p
} else {
```

```
<!DOCTYPE html PU
<html xmlns="http
  <head>
    <title>Hello
```

- 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 sends result across the network
  - script produces output that becomes the response sent back

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 locally on EasyPHP and on most commercial web hosts
- **well-documented:** type `php.net/`*functionName* in browser Address bar to get docs for any function

Hello, World!

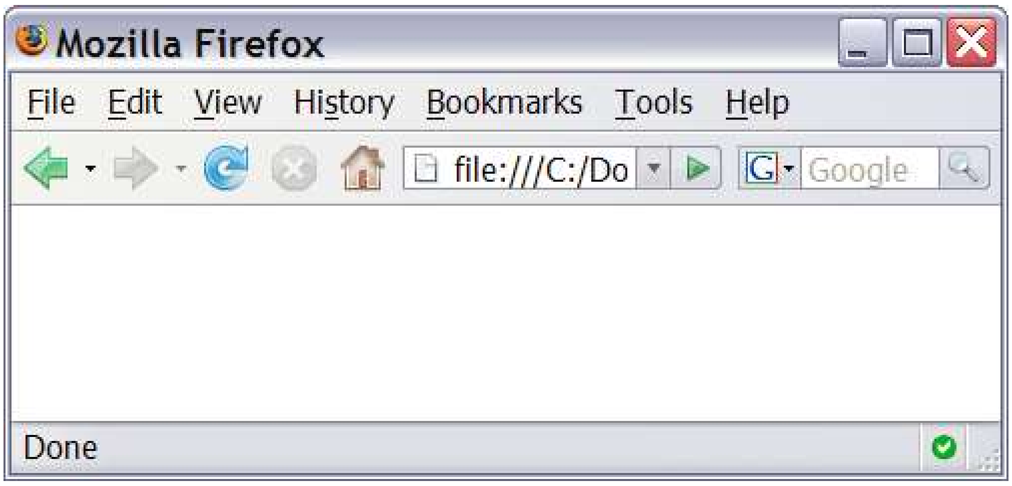
The following contents could go into a file `hello.php`:

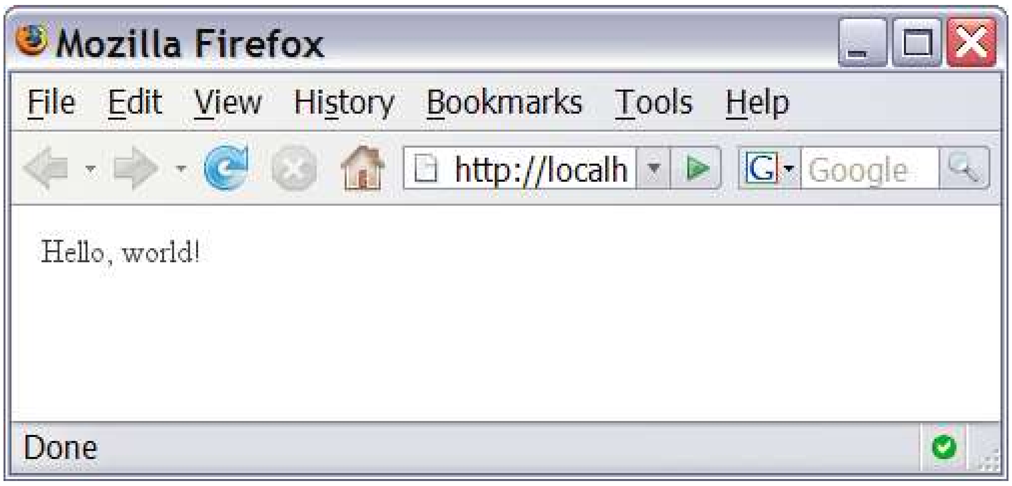
```
<?php
print "Hello, world!";
?>
```

```
Hello, world!
```

- a block or file of PHP code begins with `<?php` and ends with `?>`
- PHP statements, function declarations, etc. appear between these endpoints

Viewing PHP output





```
Hello, world!
```

- you can't view your .php page on your local hard drive; you'll either see nothing or see the PHP source code
- if you upload the file to a PHP-enabled web server, requesting the .php file will run the program and send you back its output

6.2: PHP Basic Syntax

- 6.1: Server-Side Basics
- **6.2: PHP Basic Syntax**

Console output: `print`

```
print "text";
```

```
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!';
```

```
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!
```

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

Arithmetic operators

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

Variables

```
$name = expression;
```

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

- 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";`

Comments

```
#  single-line comment

//  single-line comment

/*
multi-line comment
*/
```

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

for loop

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

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

`if/else` statement

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

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

`while` loop (same as Java)

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

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

- `break` and `continue` keywords also behave as in Java