

1

Server side basics

Be careful

2

- ❑ Do not type any command starting with “sudo” into a terminal attached to a university computer.
- ❑ You have complete control over you AWS server, just as you have complete control over your laptop.
- ❑ University computers are shared—“sudo” overrides protection of other users of the computer.XS

SM: Initialize Git on Betaweb

3

```
AWS$ pwd
```

```
/home/ec2-user/
```

```
AWS$ git clone <you>@betaweb.csug.rochester.edu:/p/  
csc210/<your team repo>
```

```
AWS$ cd <your team repo>
```

```
AWS$ cp -r /var/www/html/* .
```

```
AWS$ git add .
```

```
AWS$ git status
```

```
AWS$ git commit
```

```
AWS$ git push origin master
```

Send email to team telling them that you have set up
the master directory on Betaweb

Everyone: Getting Betaweb to AWS

4

```
AWS> pwd
```

```
/home/ec2-user/
```

```
AWS> git clone <you>@betaweb.csug.rochester.edu:/p/  
csc210/<your team repo>
```

```
AWS> cd <your team repo>
```

```
AWS> ls
```

Is everything there?

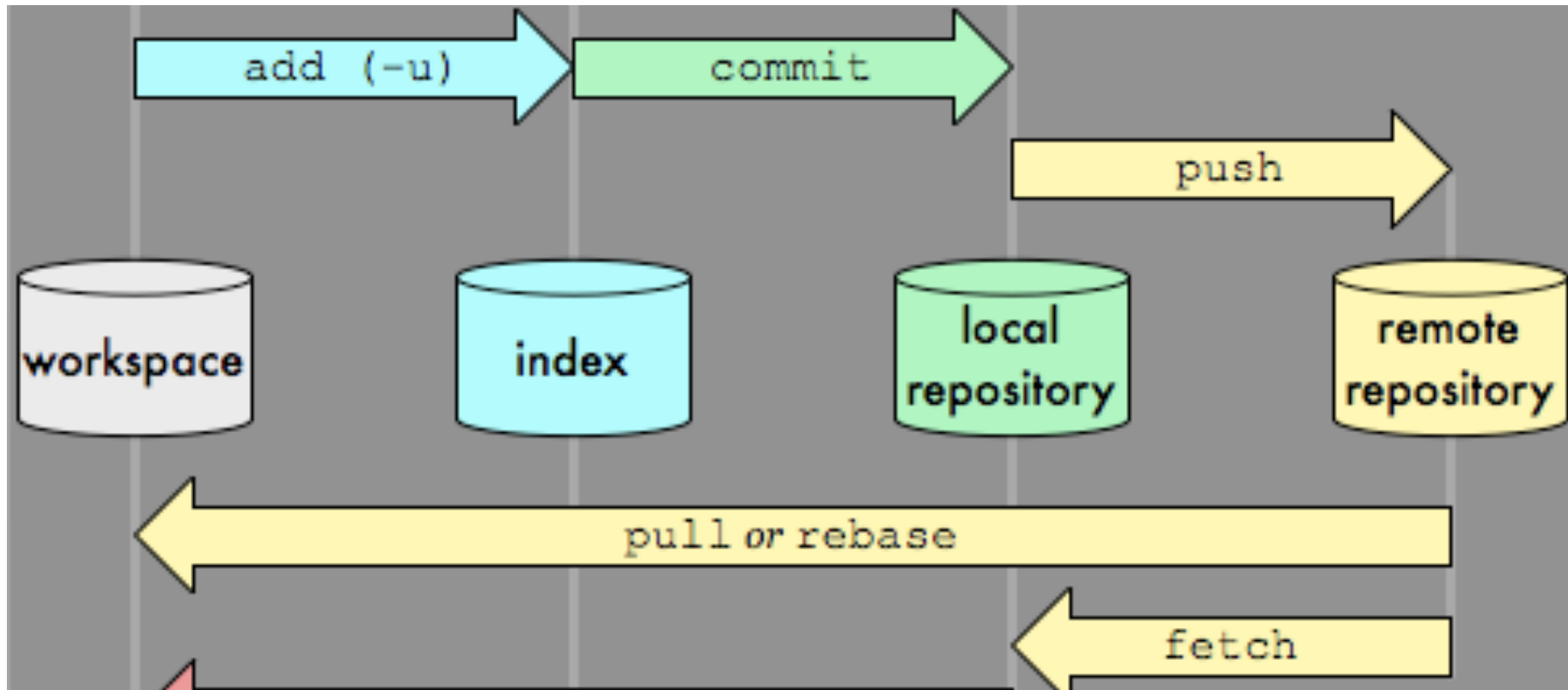
Everyone: Create personal branch

5

```
$ git branch
* master
$ git branch martin
$ git branch
  martin
* master
$ git checkout martin
Switched to branch 'martin'
$ git branch
* martin
  master
```

Git diagram

6



Everyone: Editing a file

7

```
$ cd 0
```

```
$ emacs index.html
```

```
$ cat index.html
```

```
...
```

```
<p>
```

```
(describe what you did, including any extra credit,  
here.)
```

```
Added this to my branch on AWS
```

```
</p>
```

```
...
```

Everyone: Checking in work

8

```
$ git add index.html
$ git status
# On branch martin
# Changes to be committed:
...
$ git commit
$ git push origin martin
martin@betaweb.csug.rochester.edu's password:
```


SM: Setting up as SM

9

```
betaweb$ pwd
/home/<scrum master's home>
betaweb $ git clone /p/csc210/<your team repo>
betaweb $ git branch
* master
betaweb $ git branch staging
betaweb $ git branch -a
* master
  staging
  remotes/origin/HEAD -> origin/master
  remotes/origin/martin
  remotes/origin/master
```

SM: Merging work

10

```
$ git checkout <branch with new work>
$ git pull origin martin
$ git checkout staging
$ git merge martin
$ git push origin staging
```

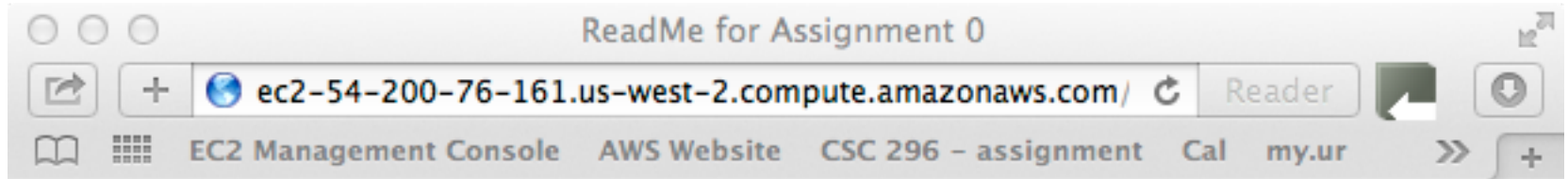
SM: Testing new work in staging

11

```
$ pwd  
/home/ec2-user/<your team repo>  
$ git branch staging  
$ git checkout staging  
$ git pull origin staging  
$ cp -r ./* /var/www/html/
```

SM: Testing new work

12



ReadMe for Assignment 0

Assignment 0

(describe what you did, including any extra credit, here.) Added this to my branch on AWS

SM: Testing new work in staging

13

```
$ pwd
```

```
/home/ec2-user/<your team repo>
```

```
$ emacs index.html
```

```
...
```

```
<p>
```

```
(describe what you did, including any extra credit,  
here.)
```

```
<p>
```

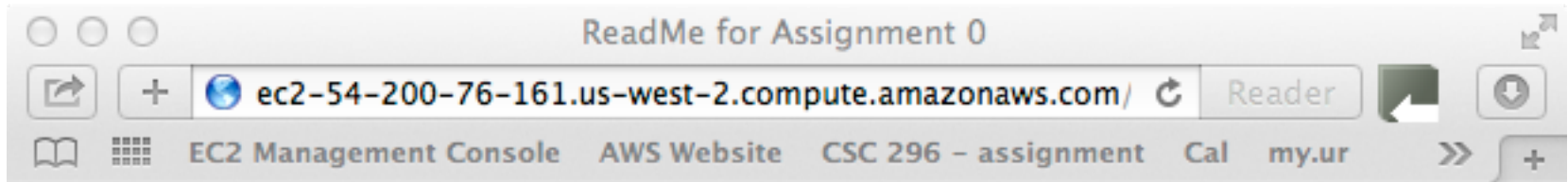
```
</p>
```

```
Added this to my branch on AWS
```

```
</p>
```

SM: Testing new work in staging

14



ReadMe for Assignment 0

[Assignment 0](#)

(describe what you did, including any extra credit, here.)

Added this to my branch on AWS

SM: Merging into Master

15

```
$ git add index.html
$ git commit
$ git push origin staging
<user>@betaweb .. Password:
$ git checkout master
$ git merge staging
$ git push origin master
Betaweb password:
```

Send email to team saying that the Master has been updated

Quiz

16

- Write CSS code put a paragraph: in the horizontal center of the page, occupying half the page's width, and right-aligning the paragraph's text?
- What is the difference between “display: none” and visibility: “hidden”
- Is PHP code static or dynamic web content?
- Given `<?php $bar = “foo” ? >` write a php block that will put “foo” on the page.

Quiz Answers

17

1.

```
p {  
    margin-left: auto;  
    margin-right: auto;  
    width: 50%;  
}
```
2. “display: none” leaves no space; “visibility: hidden” leaves space.
3. PHP is dynamic web content?
4. `<?php $bar ?>`

18

Server side basics

URLs and web servers

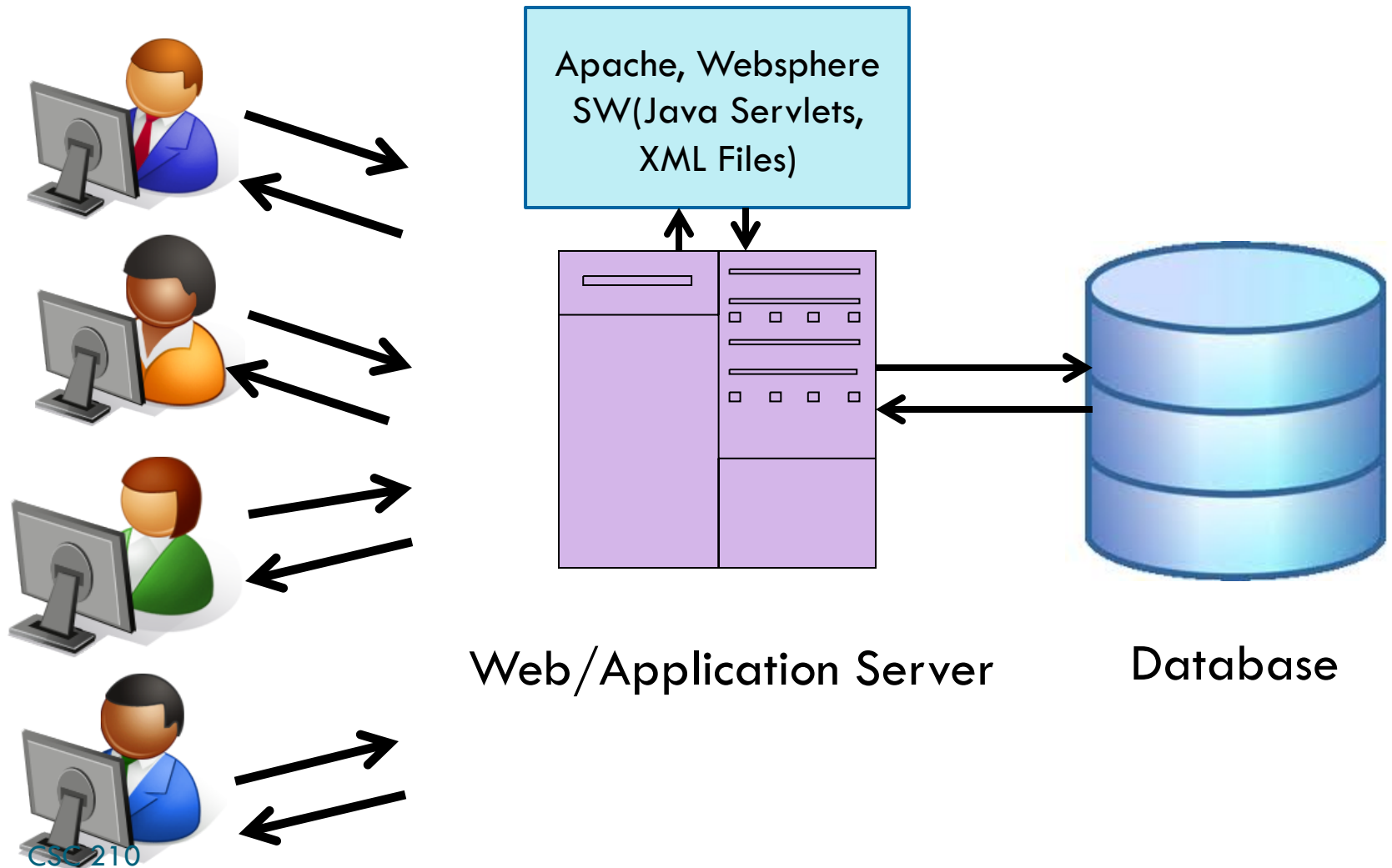
19

```
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
 - ▣ The server sends back its contents to you

URLs and web servers (cont.)

20



URLs and web servers (cont.)

21

```
http://www.facebook.com/home.php
```

- Some URLs actually specify programs that the web server should *run*, and then send their output back to you as the result:
 - ▣ The above URL tells the server **facebook.com** to run the program **home.php** and send back its output

Server-Side web programming

22

- 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



Server-Side web programming (cont.)

23

- Also called *server side scripting*:
 - Dynamically edit, change or add any content to a Web page
 - Respond to user queries or data submitted from HTML forms
 - Access any data or databases and return the results to a browser
 - Customize a Web page to make it more useful for individual users
 - Provide security since your server code cannot be viewed from a browser

Server-Side web programming (cont.)

24

- Web server:
 - ▣ contains software that allows it to run server side programs
 - ▣ sends back their output as responses to web requests
- Each language/framework has its pros and cons
 - ▣ we use PHP

What is PHP?

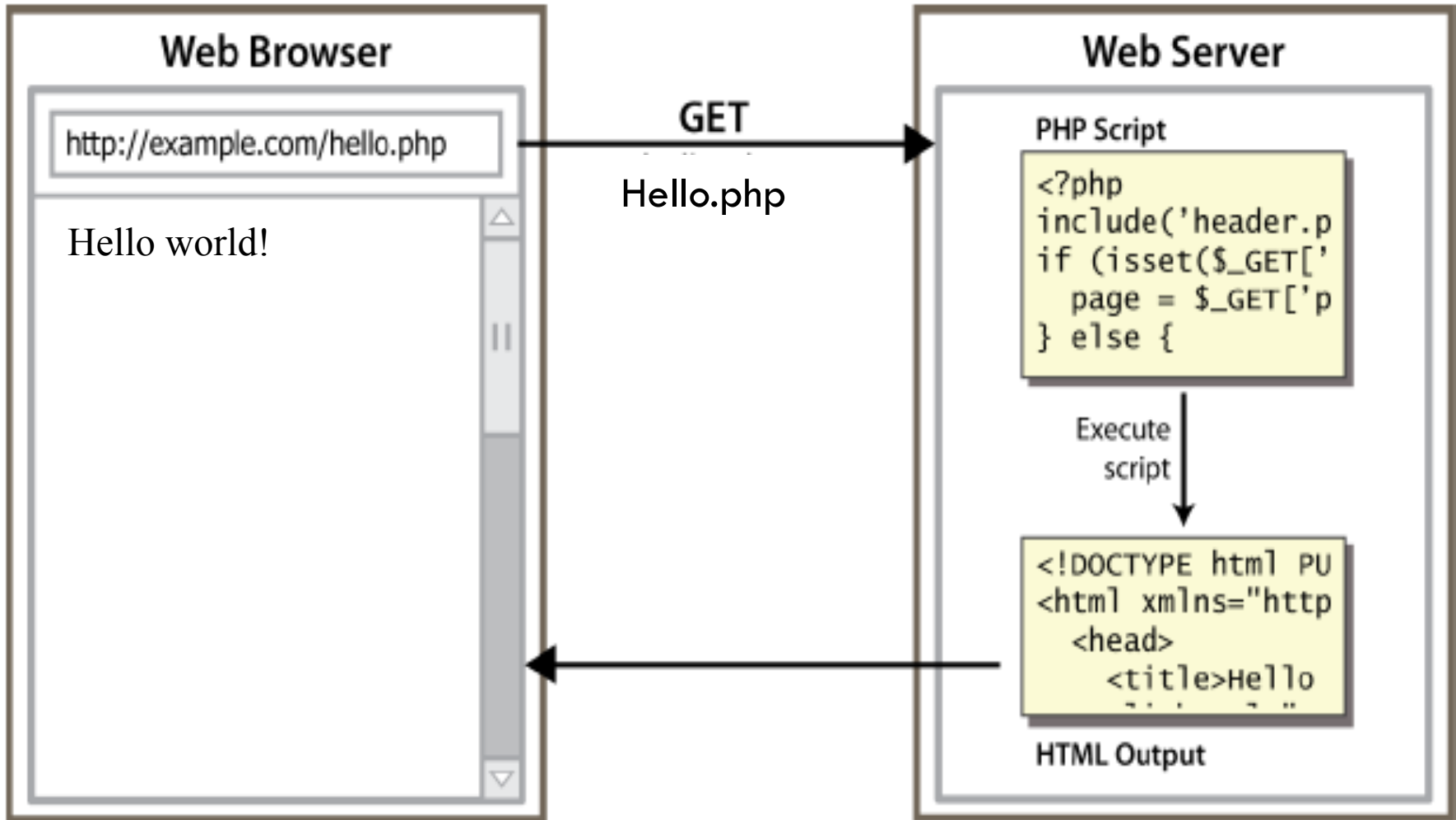
25

- PHP stands for "PHP Hypertext Preprocessor"
- 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



Lifecycle of a PHP web request

26



Why PHP?

27

- Free and open source
- Compatible
 - ▣ as of November 2006, there were more than 19 million websites (domain names) using PHP.
- Simple

Hello World!

28

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

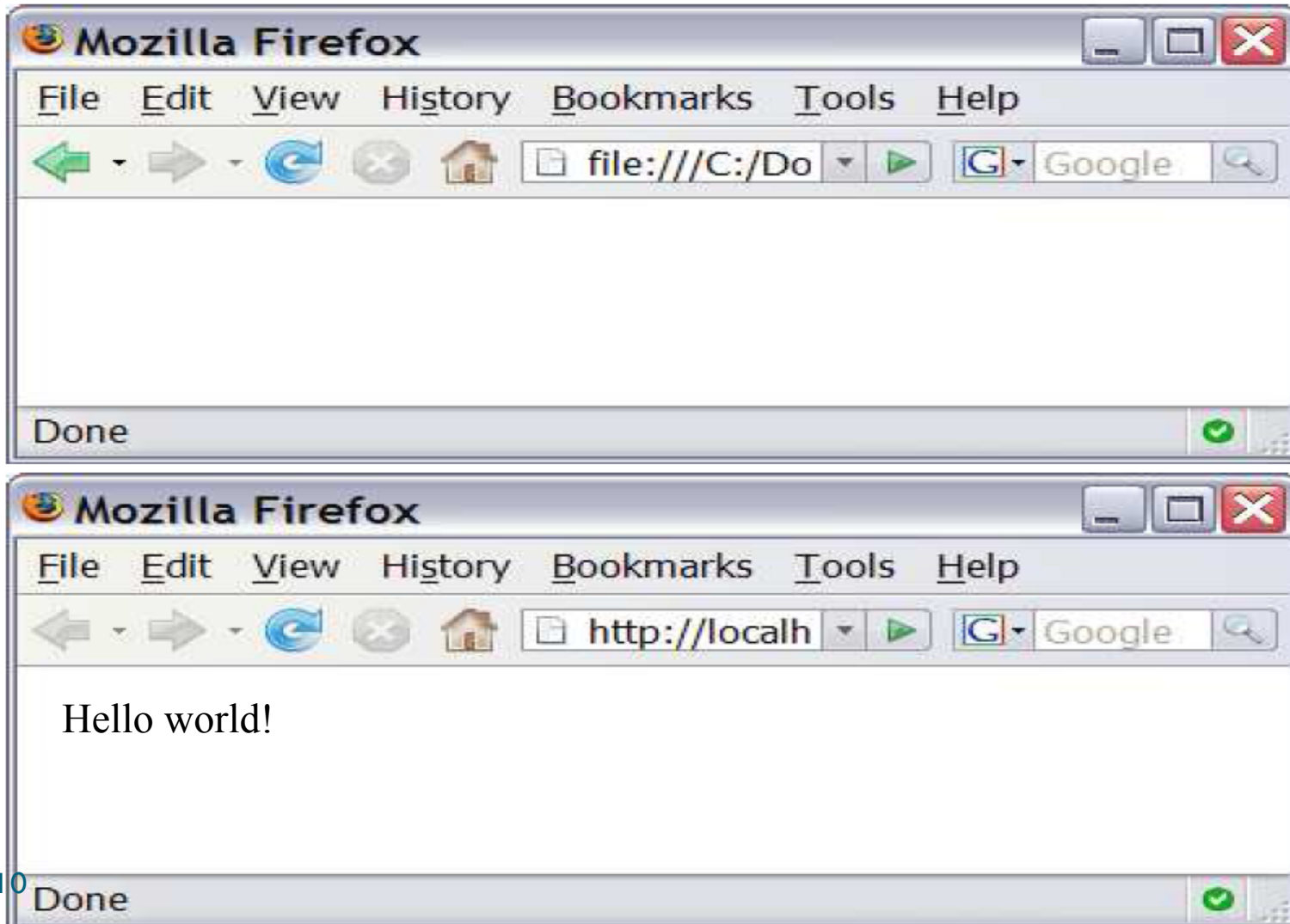
PHP

Hello world!

output

Viewing PHP output

29



30

PHP Basic Syntax

PHP syntax template

31

HTML content

<?php

PHP code

?>

HTML content

<?php

PHP code

?>

HTML content ...

PHP

- Contents of a .php file between `<?php` and `?>` are executed as PHP code
- All other contents are output as pure HTML
- We can switch back and forth between HTML and PHP "modes"

Console output: `print`

32

```
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

Variables

33

```
$name = expression;
```

PHP

```
$user_name = "mundruid78";  
$age = 16;  
$drinking_age = $age + 5;  
$this_class_rocks = TRUE;
```

PHP

- ❑ names are case sensitive
- ❑ 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)

Variables

34

- basic types: *int*, *float*, *boolean*, *string*, *array*, *object*, *NULL*
 - ▣ test type of variable with `is_type` functions, e.g. `is_string`
 - ▣ `gettype` function returns a variable's type as a 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";` CSC 210

Arithmetic operators

35

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

Comments

36

```
# 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 **//**

String Type

37

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

PHP

- zero-based indexing using bracket notation
- there is no char type; each letter is itself a String
- string concatenation operator is . (period), not +
 - `5 + "2 turtle doves" === 7`
 - `5 . "2 turtle doves" === "52 turtle doves"`
- can be specified with `""` or `"`

String Functions

38

```
# index 0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);
$cmp = strcmp($name, "Brian Le");
$index = strpos($name, "e");
$first = substr($name, 9, 5);
$name = strtoupper($name);
```

PHP

String Functions (cont.)

39

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
<u>strcmp</u>	compareTo

Interpreted Strings

40

```
$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
```


Interpreted Strings (cont.)

41

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

PHP

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

Interpreted Strings (cont.)

42

```
$name = "Xenia";  
$name = NULL;  
if (isset($name)) {  
print "This line isn't going to be reached.\n";  
}
```

PHP

- a variable is NULL if
 - ▣ it has not been set to any value (undefined variables)
 - ▣ it has been assigned the constant NULL
 - ▣ it has been deleted using the unset function
- can test if a variable is NULL using the isset function
- NULL prints as an empty string (no output)

for loop (same as Java)

43

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

PHP

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

PHP

bool (Boolean) type

44

```
$feels_like_summer = FALSE;  
$php_is_great = TRUE;  
$student_count = 7;  
$nonzero = (bool) $student_count; # TRUE
```

PHP

- 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
- FALSE prints as an empty string (no output); TRUE prints as a 1

if/else statement

45

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

PHP

while loop (same as Java)

46

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

PHP

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

PHP

Math operations

47

```
$a = 3;  
$b = 4;  
$c = sqrt(pow($a, 2) + pow($b, 2));
```

PHP

math functions

<u>abs</u>	<u>ceil</u>	<u>cos</u>	<u>floor</u>	<u>log</u>	<u>log10</u>	<u>max</u>
<u>min</u>	<u>pow</u>	<u>rand</u>	<u>round</u>	<u>sin</u>	<u>sqrt</u>	<u>tan</u>

math constants

M_PI	M_E	M_LN2
------	-----	-------

Int and Float Types

48

```
$a = 7 / 2; # float: 3.5  
$b = (int) $a; # int: 3  
$c = round($a); # float: 4.0  
$d = "123"; # string: "123"  
$e = (int) $d; # int: 123
```

PHP

- int for integers and float for reals
- division between two int values can produce a float

PHP exercise 1

49

- For your first PHP exercise, echo the following statement to the browser:

“Twinkle, Twinkle little star.”

- Next, create two variables, one for the word “Twinkle” and one for the word “star”. Echo the statement to the browser, this time substituting the variables for the relevant words. Change the value of each variable to whatever you like, and echo the statement a third time. Remember to include code to show your statements on different lines.

PHP exercise 2

50

- PHP includes all the standard arithmetic operators. For this PHP exercise, you will use them along with variables to print equations to the browser. In your script, create the following variables:

```
$x=10;
```

```
$y=7;
```

- Write code to print out the following:

```
10 + 7 = 17
```

```
10 - 7 = 3
```

```
10 * 7 = 70
```

```
10 / 7 = 1.4285714285714
```

```
10 % 7 = 3
```

- Use numbers only in the above variable assignments, not in the echo statements.

PHP exercise 3

51

- Arithmetic-assignment operators perform an arithmetic operation on the variable at the same time as assigning a new value. For this PHP exercise, write a script to reproduce the output below. Manipulate only one variable using no simple arithmetic operators to produce the values given in the statements.

- Hint: In the script each statement ends with "Value is now \$variable."

Value is now 8.

Add 2. Value is now 10.

Subtract 4. Value is now 6.

Multiply by 5. Value is now 30.

Divide by 3. Value is now 10.

Increment value by one. Value is now 11.

Decrement value by one. Value is now 10.

PHP exercise 4

52

- When you are writing scripts, you will often need to see exactly what is inside your variables. For this PHP exercise, think of the ways you can do that, then write a script that outputs the following, using the echo statement only for line breaks.

```
string(5) "Harry"
```

```
Harry
```

```
int(28)
```

```
NULL
```

PHP exercise 5

53

- For this PHP exercise, write a script using the following variable:
`$around="around";`
- Single quotes and double quotes don't work the same way in PHP. Using single quotes (' ') and the concatenation operator, echo the following to the browser, using the variable you created:
What goes around, comes around.

PHP exercise 5

54

- In this PHP exercise, you will use a conditional statement to determine what gets printed to the browser. Write a script that gets the current month and prints one of the following responses, depending on whether it's August or not:

It's August, so it's really hot.

Not August, so at least not in the peak of the heat.

- Hint: the function to get the current month is `'date('F', time())'` for the month's full name.

PHP exercise 6

55

- Loops are very useful in creating lists and tables. In this PHP exercise, you will use a loop to create a list of equations for squares.
- Using a for loop, write a script that will send to the browser a list of squares for the numbers 1-12. Use the format, "1 * 1 = 1", and be sure to include code to print each formula on a different line.

PHP exercise 7

56

- HTML tables involve a lot of repetitive coding - a perfect place to use for loops. You can do even more if you nest the for loops.
- In this PHP exercise, use two for loops, one nested inside another. Create the following multiplication table:

1	2	3	4	5	6	7
2	4	6	8	10	12	14
3	6	9	12	15	18	21
4	8	12	16	20	24	28
5	10	15	20	25	30	35
6	12	18	24	30	36	42
7	14	21	28	35	42	49