1

Server side basics

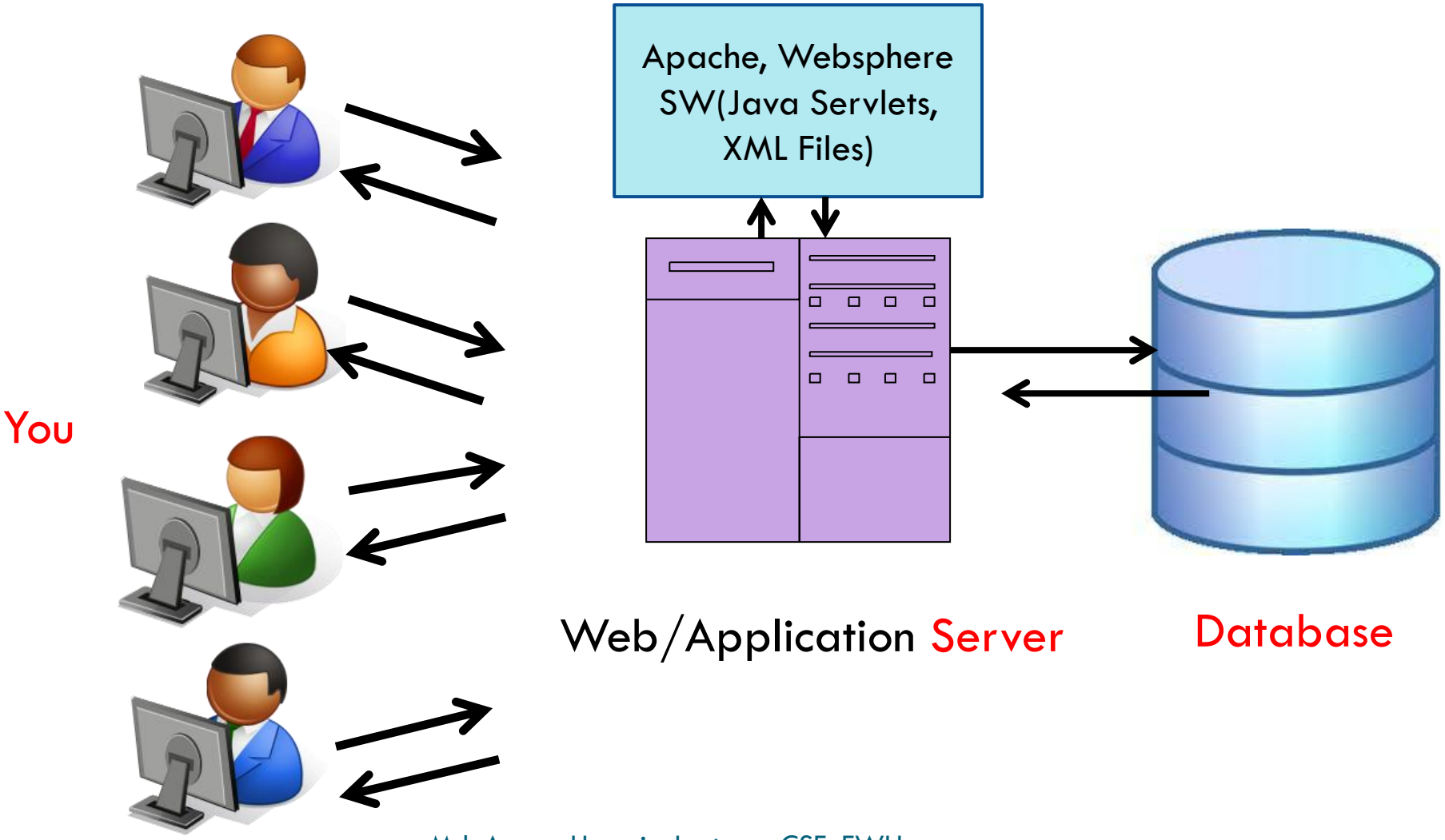
URLs and web servers

http://server/path/file

Example: http://**www.facebook.com/home.php**

- 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.)



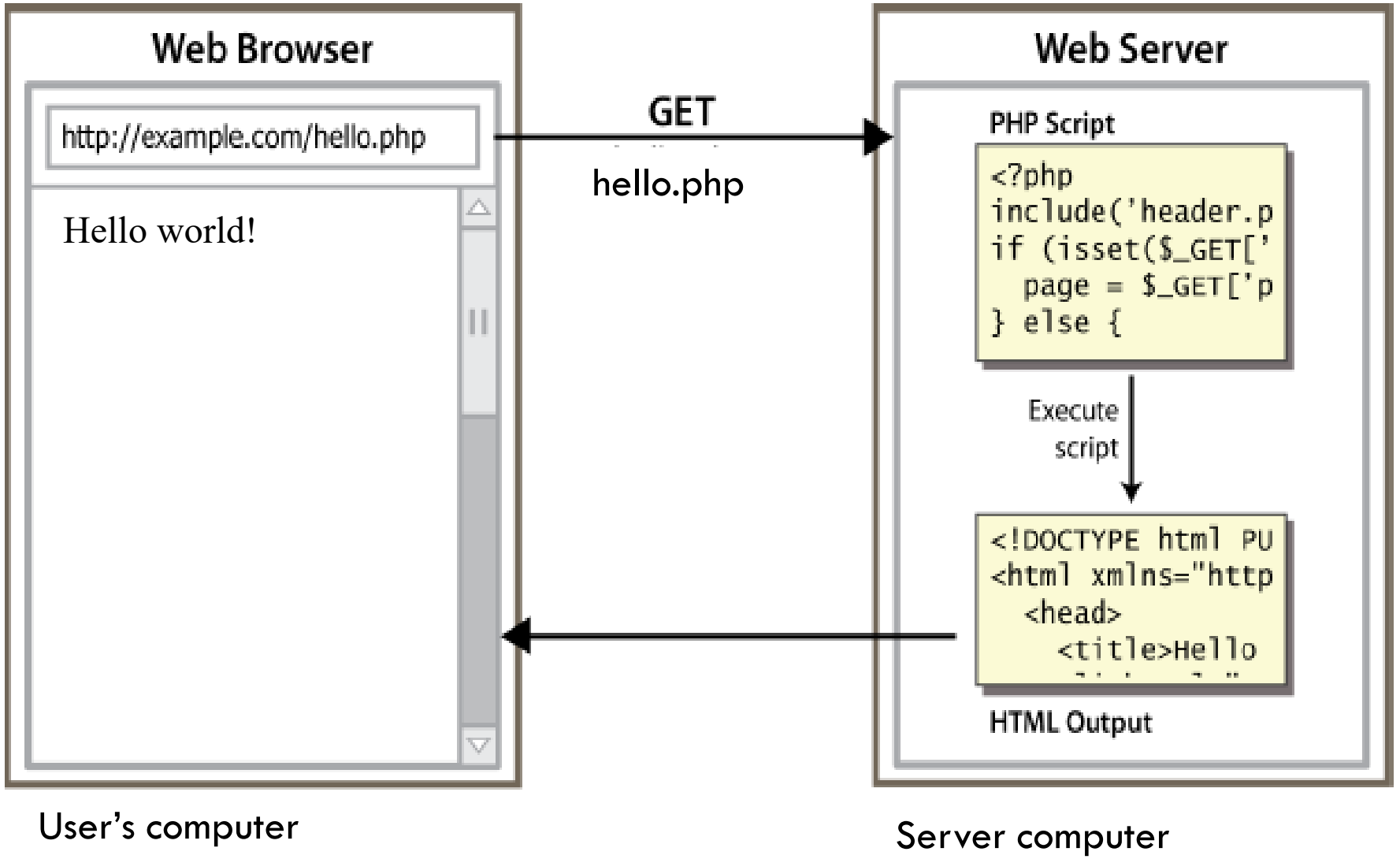
Server-Side programming

- Server side programming/scripting can:
  - Can create, destroy and manage user sessions.
  - Final validation of submitted HTML form.
  - Can access and modify files or databases.
  - Customize a Web page to make it more useful for individual users (roles).
  - Provide security since your server code cannot be viewed from a browser

Server-Side web programming

- Web server:
  - contains software that allows it to run server-side programs
  - sends back their output as responses to web requests
- 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 etc.
- Each language/framework has its pros and cons
  - we use opensource PHP (PHP: Hypertext Preprocessor) here.

Lifecycle of a PHP web request



PHP syntax template


```
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"

Output: `print/echo`


```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!.';
echo "Hello", " ", "World!";
```

PHP

output

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!. Hello World!

- HTML ignores empty space or newline. Therefore use
- Print can not take multiple parameter, echo can.
- Print act as function and have return type, echo doesn't have any.

Variable name


```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)
- Therefore, PHP is a loosely typed language (like JavaScript or Python)

Data types

- basic types: *int, float, boolean, string, array, object, NULL*
  - test type of variable with `is_*` functions, e.g. `is_string($var)`
  - `Gettype($var)` function returns a variable's type as a string
- Examples:
  - echo(gettype("PHP")); // string
  - echo(is_int(50)); // 1 (true)
  - print is_bool(True); // 1 (true)
  - echo is_array([1, 2, 3]); // 1 (true)
  - echo gettype(true); // boolean

The "+" and "/" Operators

- PHP *converts between types automatically* in many cases:
  - string → int auto-conversion on +
  - int → float auto-conversion on /
  - Examples:
    - echo "50" + "20"; // 70
    - echo "50 apples" + "20 bananas"; // 70
    - $result3 = "10" / 4; // 2.5
    - echo "hello" + "world"; // 0
- type-cast with **(type)**:
  - `$age = (int) "21";`

String


```php
$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"`

String Functions

Name	Java Equivalent
strlen	length
Strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
explode, implode	split, join
strcmp	compareTo

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

Interpreted Strings


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

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

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

Interpreted Strings (cont.)


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

PHP

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

Null Value


```php
$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)

`bool` (Boolean) type


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

- the following values are considered to be FALSE (all others are TRUE):
  - 0
  - "", and NULL (includes unset variables)
  - arrays with 0 elements
- FALSE prints as an empty string (no output);
- TRUE prints as a 1

Arrays


```php
$name = array();          # create
$name = array(value0, value1, ..., valueN); # create
$name[index]              # get element value
$name[index] = value;     # set element value
$name[] = value;          # append
```

PHP

```php
Example:
$a = array();      # empty array (length 0)
$a[0] = 23;        # stores 23 at index 0 (length 1)
$a2[] = "Ooh!";    # add string to end (at index 1)
```

PHP

- Append: use bracket notation without specifying an index
- Element type is not specified; can have mix types

for loop & for each loop


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

```
foreach ($array as $value) {
      // statements for each $value
}
foreach ($array as $key => $value) {
      // Statements for each $key and $value
}
```

```
$person = ["name" => "Alice", "age" => 25, "job" =>
"Engineer"];

foreach ($person as $key => $value) {
echo "$key: $value\n";
}
```

While loop & Condition


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

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

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

Math operations


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

PHP

math functions

abs	ceil	cos	floor	log	log10	max
min	pow	rand	round	sin	sqrt	tan

math constants

M_PI	M_E	M_LN2

Functions


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

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

- parameter types and return types are not written
- a function with no return statements implicitly returns NULL