PHP

Functions, Files, Classes, AND objects

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

VIDEO

https://www.youtube.com/watch?v=a7_WFUlFS94

PHP – OVERVIEW

- PHP is a dynamically typed scripting language
- PHP natively provides more functionality than JS
    - Most notably classes and other OO principles

PHP – INCLUDE

- PHP code can be distributed over different files
- Scripts can then import other scripts (with a relative path) and use their code

- include
  - Try to import the given script
  - If the import fails, throw a warning and continue executing
- require
  - Try to import the given script
  - If the import fails, throw a fatal error and stop executing

PHP – INCLUDE

- If a script is imported multiple times
  - PHP will throw an error
  - Now there are duplicate imports of variables, functions, and classes
  - PHP does not know which of those duplicates to call

- Two solutions
  - Wrap each variable, function, class inside and exists statement (eg: function_exists) which requires a lot more coding and is generally not a good idea
  - Alternatively import scripts only once, PHP will check if the script was already imported
    - include_once
    - require_once

PHP – INCLUDE

```php
<?php
       // Might be included multiple times
       include "utils.php";
       require("utils.php");

       // Only included once
       include_once("utils.php");
       require_once "utils.php";

       // Include with absolute paths
       // Current script's dir
       include(dirname(__FILE__) . DIRECTORY_SEPARATOR . "utils.php");
       // Current script's dir
       require __DIR__ . DIRECTORY_SEPARATOR . "utils.php";
       // Root dir
       require_once $_SERVER["DOCUMENT_ROOT"] . "utils.php"; ?>
```

PHP – FUNCTIONS

- Define global functions

```php
<?php
	function sayHello()
	{
		echo "I say hello";
	}

	sayHello(); // Call the function
?>
```

PHP – FUNCTIONS

- Define functions with parameters and return values
- Since PHP is dynamically typed, functions can return different data types

```php
<?php
	function sayHello($message)
	{
		echo $message;
		if($message == "") return null;
		else return "Yeah";
	}

	$result = sayHello("I say hello");
?>
```

PHP – FUNCTIONS

- Define functions with default parameters

```php
<?php
	function sayHello($message = "", $suffix = "!")
	{
		echo $message . $suffix;
	}

	sayHello();
	sayHello("I say hello");
	sayHello("You say goodbye", "?");
?>
```

PHP – FUNCTIONS

- Note that the following is possible in other scripting languages (eg: Python)

```
# Not providing a value for the first parameter
sayHello(suffix = "...");
```

- Named parameters are not possible in PHP
- All parameters before the one of interest have to be manually provided

```
<?php
        // Manually provide the default value for $message
        sayHello("", "...");
?>
```

PHP – FUNCTIONS

```php
<?php
        function sum($values)
        {
                $sum = 0;
                foreach($values as $value)
                {
                        $sum += $value;
                }
                return $sum;
        }

        $array = [45, 98, 52, 30];
        echo "The sum is: " . sum($array);
?>
```

PHP – FUNCTIONS

- Parameters can also be passed by reference and updated inside the function

```
<?php
        function sum($values, &$sum)
        {
                if(count($values) == 0) return false;
                foreach($values as $value)
                        $sum += $value;
                return true;
        }

        $array = [45, 98, 52, 30];
        $sum = 10;
        if(sum($array, $sum)) echo "The sum is: " . $sum;
        else echo "Empty array";
?>
```

PHP – FILES

- Easily read all data from a file

```
<?php
	echo readfile("data.txt");
?>
```

PHP – FILES

- Opening, reading, and closing files

```
<?php
	$file = fopen("data.txt", "r") or die("File could not be opened");
	$size = filesize("data.txt");
	echo fread($file, $size);
	fclose($file);
?>
```

PHP – FILES

- Read line by line

```
<?php
	$file = fopen("data.txt", "r") or die("File could not be opened");
	while(!feof($file))
	{
		echo fgets($file) . "<br>";
	}
	fclose($file);
?>
```

PHP – FILES

- Write to a file
- If the file does not exist, it will be created
- If the file exists, it will be overwritten
- Use flags for different file handling (eg: "a" for append)

```php
<?php
	$file = fopen("data.txt", "w") or die("File could not be opened");
	fwrite($file, "Bitcoin\n");
	fwrite($file, 10256);
	fclose($file);
?>
```

PHP – CLASSES

- Create custom classes
- Default visibility is public (in C++ they are private by default)

```
<?php
        class Crypto
        {
                private $symbol = "";
                private $price = 0;
                public function display() // public member function
                {
                        echo $this->symbol . ": " . $this->price;
                }
        }

        $crypto = new Crypto();
        $crypto->display();
?>
```

PHP – CLASSES

- Constants (do not change value) and static (instantiated once per class)

```
<?php
        class Crypto
        {
                const FIXED = "variable does not change";
                static $static = "one instance per class";

                public static function display()
                {
                        echo self::FIXED . self::$static;
                }
        }

        Crypto::display();
        echo Crypto::FIXED . Crypto::$static;
?>
```

PHP – CLASSES

- Create custom classes

```php
<?php
        class Crypto
        {
                // private member variable
                private $symbol = "";
                private $price = 0;
                public function display() // public member function
                {
                        echo $this->symbol . ": " . $this->price;
                }
        }

        $crypto = new Crypto();
        $crypto->display();
?>
```

PHP – CLASSES

- Use constructors and destructors (starts with double underscore)

```php
<?php
        class Crypto
        {
                private $symbol = "";
                private $price = 0;

                function __construct($symbol, $price = 0)
                {
                        $this->symbol = $symbol;
                        $this->price = $price;
                }
                function __destruct(){ }
        }

        $bitcoin = new Crypto("BTC", 10245);
?>
```

PHP – CLASSES

- Class inheritance

```
<?php
	class Bitcoin extends Crypto
	{
		const SYMBOL = "BTC";

		function __construct($price = 0)
		{
			parent::__construct(self::SYMBOL, $price);
		}
	}

	$bitcoin = new Bitcoin(10245);
?>
```

PHP – SINGLETON

- Singleton is a design pattern (COS214)
- Singleton allows you to only create a single instance of a class
- All future instantiations of that class will refer back to your original instance

- Many applications in web development
  - Database: all queries are managed by the same instance (only 1 connection required)
  - Configurations: retrieve server configurations through a singleton instance
  - Global variables: Handle access to globals (eg: $_GET, $_POST, $_SESSION)

PHP – SINGLETON

- Singleton should restrict the programmer to create new instances
  - Make the constructor private or protected
  - Provide a static member function to create the instance
  - Always keep the destructor public, since PHP will call it if the instance runs out of scope

PHP – SINGLETON

```php
<?php
   class Database
   {
      public static function instance()
      {
         static $instance = null;
         if($instance === null) $instance = new Database();
         return $instance;
      }

      private function __construct() { // Connect to the database }
      public function __destruct() { // Disconnect from the database }
      public function retrieveUser($username){ // Retrieve from the database }
   }

   $user = Database::instance()->retrieveUser("satoshi");
?>
```

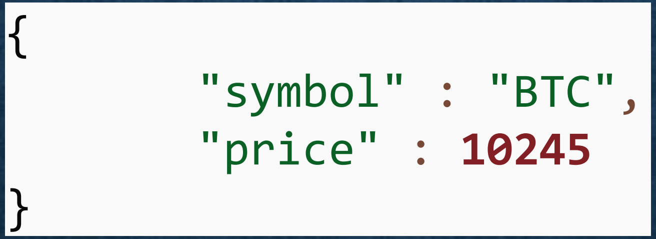
PHP – OBJECTS

- Objects can be created with
  - Built-in classes/types
  - Custom defined classes
  - PHP standard class

- The standard class is a generic empty class
- Properties can be dynamically added

PHP – OBJECTS

- Given the following JSON object



```
{
        "symbol" : "BTC",
        "price" : 10245
}
```

PHP – OBJECTS

```
{
	"symbol" : "BTC",
	"price" : 10245
}
```

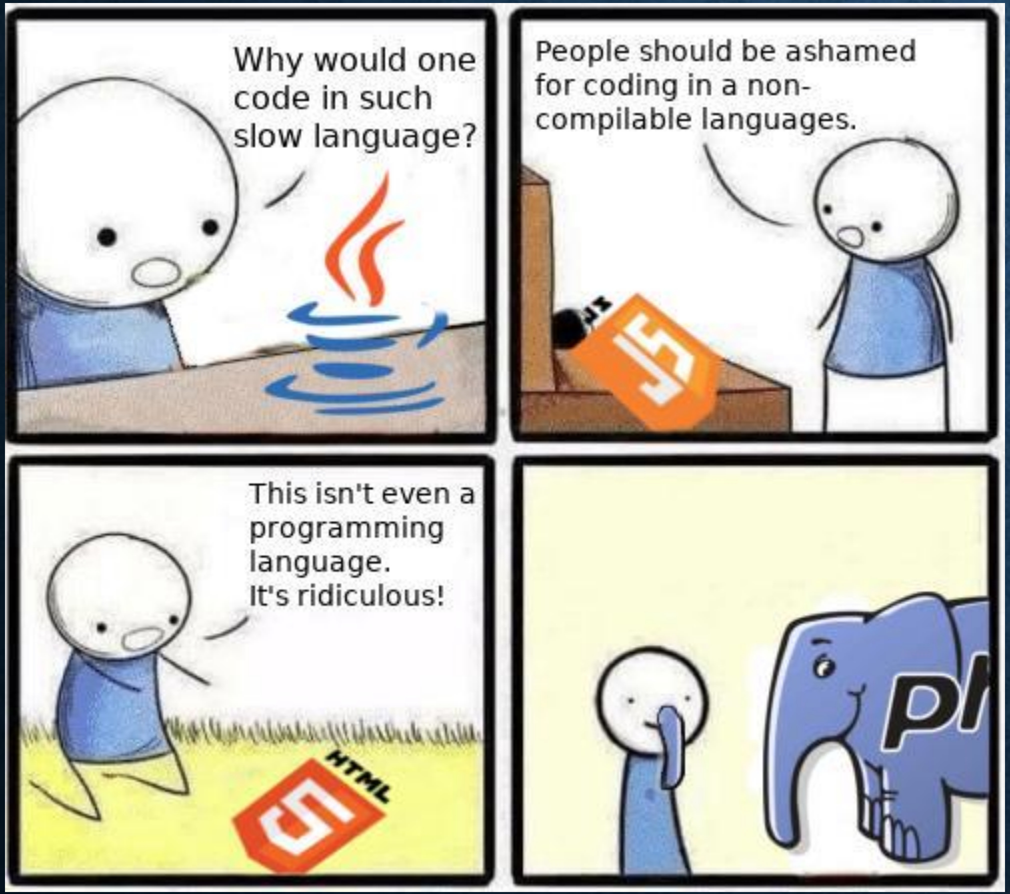
- Create the equivalent JSON object in PHP

```php
<?php
	$object = new stdClass();
	$object->symbol = "BTC";
	$object->price = 10245;
?>
```

- Or equivalent using associative arrays

```php
<?php
	$object = (object) [
		"symbol" => "BTC",
		"price" =>10245
	];
?>
```

MEMES


Why would one code in such slow language?
People should be ashamed for coding in a non-compilable languages.
JS
This isn't even a programming language. It's ridiculous!
HTML
PH

C++ pointers when you forget to clean them up
fb.me/objcmemes
I'm not supposed to exist that long. This is getting weird.