Web Technologies — Week 11

Mikheil Rukhaia

International Black Sea University,
Tbilisi, Georgia

December 17, 2015

Outline

Introduction to PHP

History

- Rasmus Lerdorf in 1994 created a set of tool on C++ to replace Perl scripts on his web page.
- The original name PHP comes from Personal Home Page.
- PHP was released for the public in 1995 as PHP 2.0.
- Nowadays, the official name is PHP: Hypertext Preprocessor and recently released version is 5.6.

What is PHP?

- PHP is a server-side scripting language.
- PHP is an interpreted language.
- PHP script runs on the server and when it is done, generated HTML page is sent to the client.
- PHP is very powerful and can:
  - process form data,
  - read, write and create files on the server,
  - work with database, and many more.

Running PHP

- To run PHP scripts install Apache or Internet Information Services (IIS) web server and PHP on it.
- It is very common to have LAMP: Linux, Apache, MySQL, PHP, but IIS is preferable on Windows.
- For complete instruction how to install PHP on Windows IIS see `http://www.php.net/manual/en/install.windows.iis7.php`
- Web interpreter: `http://www.codecademy.com/courses/web-beginner-en-StaFQ/0/1`

Embedding in HTML

- Popularity of PHP lies on its simple integration with HTML.
- A PHP document is HTML document, where PHP code is written inside the `<?php /* code*/ ?>` tag.
- Everything written inside such tags is interpreted on server, other parts are returned to the user unchanged.

First PHP script

- Example:

```
<html>
<head>
 <title>First PHP Script</title>
 <link rel="stylesheet" type="text/css"
                        href="style.css" />
</head>
<body>
 <?php
  // Get current time and display greeting
  $time = date("D, d M Y, g:i:s a");
  echo "<h1>Hello at $time!</h1>";
 ?>
</body>
</html>
```

`echo()` and `date()`

- `echo()` send text and other displayable data types as part of HTML page.
- `print()` can be used for the same purpose, but it returns boolean value as well.
- `date()` returns the current date and the argument string tells how to format the date:
  - `D` stands for the week day and `d` form the day in month.
  - `M` stands for month name and `m` for the month number.
  - `Y` stands for full year and `y` for last two digits.
  - `G` stands for hours in full-day format and `g` in half-day format.
  - `i` and `s` stands for minutes and seconds, respectively.
  - `a` stands for either `am` or `pm`.

Variables

- Variables are declared and used exactly in the same way as in JavaScript.
- The only difference is in variable names.
- Variable names must begin with a dollar sign `$`.
- The first character after the dollar sign must be a letter or an underscore.
- The remaining characters may be letters, numbers, or underscores.
- Variable names are case-sensitive.

Data Types

Type	Description	Type Check
Integer	whole number	`is_int(value)`
Float	floating-point number	`is_float(value)`
String	series of characters	`is_string(value)`
Boolean	either true or false	`is_bool(value)`
Array	ordered map (names or numbers mapped to values)	`is_array(value)`
Object	type that may contain properties and methods	`is_object(value)`
Resource	reference to an external resource, such as a file or database	`is_resource(value)`
Null	only contains `null` as a value	`is_null(value)`

Expressions

- PHP expressions are written in the similar way as in JavaScript, but semicolon is mandatory at the end of the expression.
- One difference in operators is the concatenation operator, which is dot in PHP unlike JavaScript's plus.
- Example:

```
$name="Mikheil";
$surname="Rukhaia";
$fullname=$name." ".$surname;
```

Logical operators

- PHP considers the following values to be false:
  - The literal value `false`
  - The integer zero `0`
  - The float zero `0.0`
  - An empty string `""`
  - The string zero `"0"`
  - An array with zero elements
  - The special type `null` (including any unset variables)
  - A SimpleXML object that is created from an empty XML tag.
- All other values are considered true in a Boolean context.
- In PHP it is possible to use `and`, `or`, `xor` as logical operators, but have lower precedence than assignment operator.

Conditionals and looping

- The conditionals of PHP are very similar to the one of JavaScript.
- The only exception is `elseif` keyword, which is written together in PHP unlike JavaScript.
- Example:

```
<?php
 if ($name=="Mikheil") { echo "Hello ".$name; }
 elseif ($name=="Rukhaia") { echo "Surname"; }
 else { echo "Done" }
?>
```

- The syntax of while, do..while and for loops are the same.

Arrays

- Arrays are defined using `array()` construct.
- It is possible to create associative array using `"Key"=>"Value"` pairs.
- `foreach` cycle can be used to traverse an array.
- Example:

```
$nums = array(1,2,3,4,5);
$alph = range("a","z",2);  //a,c,e,...
$map = array("one"=>1,"two"=>2);
foreach ($map as $key=>$value)
  echo "$key = $value";
```

Strings

- String is a fundamental data type in Web.
- HTML pages, HTTP header and URLs consist of strings of plain text.
- Web programming languages such as PHP are particularly geared toward working with strings.
- There are more than 100 functions manipulating strings in PHP (for a complete list consult `php.net`)

Creating strings

- To create strings it is possible to use either single, or double quotes.
- String in single quotes is interpreted as it is.
- String in double quotes is parsed.
- Example:

```
$myString = 'world';
echo "Hi $myString!"; // Hi world!
echo 'Hi $myString!'; // Hi $myString!
echo "<pre>Hi\tthere!</pre>"; // Hi   there!
echo '<pre>Hi\tthere!</pre>'; // Hi\tthere!
```

Escape characters

Sequence	Meaning
`\n`	New-line character (ASCII 10)
`\r`	Carriage return character (ASCII 13)
`\t`	Horizontal tab character (ASCII 9)
`\v`	Vertical tab character (ASCII 11)
`\f`	New-page character (ASCII 12)
`\\`	Backslash
`\$`	$ symbol
`\"`	Double quote

Upper and lowercase

- `strtolower()` and `strtoupper()` transform strings to lowercase and uppercase, respectively.
- `lcfirst()` and `ucfirst()` transform the first letter of strings to lowercase and uppercase respectively.
- `ucwords()` transforms the first letter of each word in a string to uppercase.

Complex strings

- In PHP it is possible to wrap large strings on several lines (simply insert newlines by pressing the Enter key).
- If concatenation of strings to variable name is needed, use curly braces to avoid confusion: `echo "The {$animal}s";`
- It is possible to insert more complex variable values inside the curly braces:
  `echo "My name is {$person["name"]}";.`
- The above is equivalent to:
  `echo "My name is ".$person["name"];`

Delimiters

- It is possible to create custom delimiters for strings using heredoc and nowdoc syntax, corresponding respectively to double and single quotes.

- Example:

```
$page=<<<HOMEPAGE
 everything in here is a parsable string!
HOMEPAGE;

$page=<<<'PAGE'
 everything in here is a non-parsable string!
PAGE;
```

- Delimiters must contain only letters, numbers and underscore and must not start with number.

Writing Functions

Function syntax

- Functions are available only to the scripts in which they were declared.
- Declaration starts with `function` keyword, followed by the name of the function, its parameters and its body.
- Example:

```
function f($str) {
  echo $str;
}
```

Function syntax (ctd.)

- PHP does not support function overloading, so every function must have a unique name.
- Function name can contain letters, digits and underscore, but cannot start with digits.
- Unlike variable names, function names are NOT case sensitive.

Function syntax (ctd.)

- You can declare parameter as optional by assigning a value to it.
- Example:

```
function f($str, $nl="<br />") {
  echo $str.$nl;
}
f("test");
f("test", "no br");
```

Variable functions

- Very useful feature of PHP are variable functions and variable variables.
- If you have a function (or variable) name stored in a variable as a string, you can call the function by appending brackets (and parameters) to the variable name.
- Example:

```
$vfunc = "f";
$func = "vfunc";
$vfanc("variable function");
$$func("variable function");
```

Variable scope

- Variables declared inside a function are local variables, not visible outside.
- Variables declared outside functions are global variables, not directly visible inside functions.
- Use `$GLOBALS` array to call global variables inside functions.
- Example:

```
$br = "<br />";
function f($str) {
  echo $str.$GLOBALS['br'];
}
```

Object-Oriented PHP

Creating classes and objects

- To create a class, use `class` keyword.
- To create an object, use `new` keyword, followed by the class name.
- Example:

```
class Example {}
$e = new Example();
```

- A common coding standard is to begin a class name with a capital letter, though you do not have to do this.
- The main thing is to be consistent.

Creating properties

- There are three visibility constraints in PHP: `public`, `private`, and `protected`.
- Property declarations are similar to variable declarations, preceded by a visibility constraint.
- To access a property from an object use `->` symbol and the property name without the `$` sign.

Creating properties (ctd.)

- Example:

```
class Example {
  public $name;
  private $count = 0;
}
$e = new Example();
$e->name = "example";
echo $e->name;
echo $e->count;              // not allowed
```

Creating properties (ctd.)

- Static class properties are created in a similar way by adding the `static` keyword.
- Static properties are called from a class name using `::` and property name (including `$` sign).
- Example:

```
class MyClass {
  public static $myProperty;
}
MyClass::$myProperty = 123;
echo MyClass::$myProperty;
```

Creating constants

- Constants are created using `const` keyword.
- It is a common practice to use all uppercase letters for constants.
- Note: constants are always public and they do not need the `$` sign.
- Example:

```
class MyClass {
  const MYCONSTANT = "example";
}

echo MyClass::MYCONSTANT;
```

Creating methods

- Methods are functions inside the classes, so they are created exactly the same way as functions.
- Additionally, methods can have a visibility constraint, but if not provided `public` is assumed.
- Example:

```
class MyClass {
  function aMethod () {
    echo "in a method";
  }
}
$e = new MyClass();
$e->aMethod();
```

Accessing methods and properties

- To use methods and properties inside the class, use `$this` object.
- Note: it is not possible to use non-static properties in static methods.
- Example:

```
class MyClass {
  public $myProperty;
  function aMethod () {
    echo $this->myProperty;
  }
}
$e = new MyClass();
$e->aMethod();
```

Type hinting

- It is important to check that correct object is passed to a function.
- In case of primitive types recall the functions `is_int()`, `is_string()`, and the like.
- When it comes to arrays and classes, you can use `array` keyword or a class name before a parameter name.
- Example:

```
function myFunc(array $a) {
  echo $a[1];
}
function drive(Car $c) {
  $c->startEngine();
}
```

Inheritance

- To create a child class based on a parent class, use the `extends` keyword.
- Example:

```
class Shape {}
class Circle extends Shape {}
```

- To prevent inheritance, use `final` keyword before a class or method definition.
- Example:

```
final class Shape {}
class Circle extends Shape {}  // error
```

Inheritance (ctd.)

- To call methods from parent class in child class use `parent::` construct followed by a method name.

- Example

```
class Fruit {
  function consume() {}
}
class Ananas extends Fruit {
  function consume() {
    $this->peel();
    parent::consume();
  }
}
```

Interfaces

- PHP allows to create `abstract` classes and `interfaces` as well.
- Interfaces cannot contain property definitions and method implementations, while abstract classes can.
- To do multiple inheritance, use `implements` keyword and list the interfaces to implement.
- Example:

```
abstract class Editor {}
interface Editable {}
interface Scrollable {}
class Text extends Editor
            implements Editable, Scrollable {}
```

Constructor and destructor

- To create a constructor, simply add a method with the special name `__construct()` (two underscores).
- In the same way, for a destructor add a method with the name `__destruct()`.
- Constructor can have parameters as well, but not destructor.
- Note: constructor overloading is not possible in PHP.

Advanced Techniques

Loading files

- It is a common practice to define classes in its own `.php` files.
- It is also useful to define common functions in a separate `.php` file and use it in every document, where necessary.
- To load other script files into a document there are functions: `include()`, `require()`, `include_once()`, and `require_once()`.

Loading files (ctd.)

- All these functions are similar, but there are important differences.
- `require()` gives an error message if file not found or does not have proper permissions, while `include()` gives just a warning.
- If the same function is included twice accidentally, then error message is generated.
- `include_once()` and `require_once()` functions solve this problem simply ignoring file content if it is already loaded.

Serialization

- PHP provides functions to store (load) an object state on (from) hard drive or database.
- `serialize()` converts an object state into a string of text.
- `unserialize()` takes a string created by `serialize()` and turns it back into a usable object.

Determining object type

- To find out the class of an object, use `get_class()` function.
- Disadvantage of this function is that it cannot determine whether object is descended from a given class.
- For this purpose there is `isinstanceof` operator.
- Example:

```
if (get_class($object) == "ClassName") {}
if ($object instanceof ClassName) {}
```

Laboratory Work

Exercises

- Write a `Calculator` class that can store two values, then add them, subtract them, multiply them together, or divide them on request.
- Create another class, `CalcAdvanced`, that extends (inherits from) the `Calculator` class and has additional operations like `pow()`, `sqrt()`, and `exp()`.
- Create an HTML form that makes use of these classes.

Discussion?!