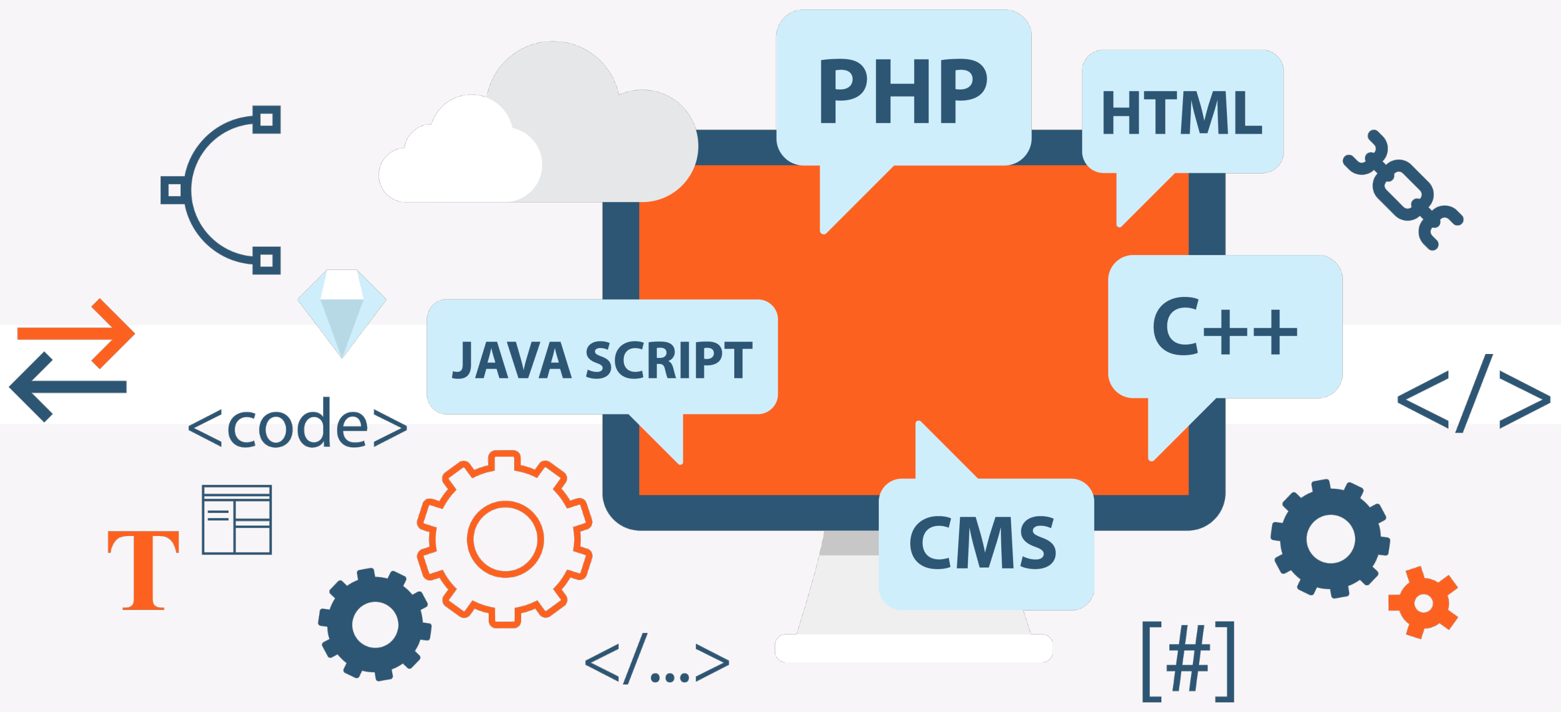
PHP

INFO3602
Web Programming & Technologies II



PHP Syntax

Week 1

Outline

- PHP
    - Variables
    - Constants
    - Arrays
    - Operators
    - Conditions
    - Loops
    - Functions

PHP Mode

Recall PHP Mode where we interleave HTML and PHP code.

```
<!DOCTYPE html>
<html>
    <head>
        <title> My Second PHP Web Page </title>
    </head>
    <body>
        <?php print ('Hello'); ?>
    </body>
</html>
```

Tip: You may wish to install an extension in Visual Studio Code that provides highlighting, smart completions with IntelliSense, and customizable formatting. Open the Extensions view (⇧⌘X on Mac, Ctrl+Shift+X on Linux and Windows) and install an appropriate plugin (usually recommended automatically).

Comments

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

```
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-line comment block
that spans over multiple
lines
*/
```

Echo and Print

With PHP, there are two basic ways to get output:

- echo
- print

```php
    <?php
        echo "this is a print <br/>";
        print "this is also a print <br/>";
        echo "this is a <b> serious </b> print <br/>";
        echo "this ", "is ", "a ", "6 ", "string ", "print <br/>";
    ?>
```

Echo and Print

The differences are small:

- `echo` has no return value while `print` has a return value of 1 so it can be used in expressions.
- `echo` can take multiple parameters (although such usage is rare) while `print` can take one argument.
- `echo` is marginally faster than `print`.

Variables

In PHP, a variable starts with the **$ sign**, followed by the name of the variable.

- A variable starts with the $ sign, followed by the name of the variable
- A variable name **must start with a letter** or the **underscore** character
- A variable name cannot start with a number
- A variable name can **only contain alpha-numeric characters** and underscores (A-z, 0-9, and _)
- Variable names are **case-sensitive** ($age and $AGE are two different variables)

Variable Types

PHP is a loosely type language and automatically associates a data type to the variable, depending on its value. Data type declarations aren’t enforced.

- Boolean
- Integer
- Floating Point
- String

```php
<?php

    $logged_in = true;   // boolean variable
    $page_number = 25;   // integer variable
    $price = 29.99;      // floating point variable
    $userName = 'lou';   // string variable

?>
```

Variable Scope

PHP has three different variable scopes:

1. local
2. global
3. static

Global and Local Scope

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```php
<?php

    function printGreeting(){
        $greeting = "Welcome";
        echo $greeting , " ";
    }

    printGreeting();

?>
```

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
    $userName = 'lou';

    function printUsername(){
        echo $userName; //this generates an error

    }
    echo $userName; // this is fine

?>
```

Global and Local Scope

The **global** keyword is used to access a global variable from within a function.

```php
<?php
    $userName = 'lou';

    function printUserName(){
        global $userName;
        echo $userName; //this is fine
    }

   printUserName();

?>
```

Static Keyword

Normally, when a function is completed/executed, all of its variables are deleted.

The `static` keyword maintains a local variable's state beyond the execution of a function.

```php
<?php

    function testStatic(){
        static $howManyTimes = 1;
        echo ($howManyTimes);
        var_dump($howManyTimes); // shows the variable type and value
        $howManyTimes++;

    }

    testStatic();
    testStatic();
    testStatic();
?>
```

Constants

A **constant** is an identifier (name) for a simple value.

The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are **automatically global** across the entire script.

```php
<?php
    define("GREETING", "This will be my greeting");
    echo GREETING;
?>
```

define( ) takes 3 Parameters:

- name: Specifies the name of the constant
- value: Specifies the value of the constant
- case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

Conditional Statements

In PHP, when we wish to perform different actions for different conditions, we can use the following conditional statements as appropriate:

- **if statement** - executes some code if one condition is true
- **if...else statemen**t - executes some code if a condition is true and another code if that condition is false
- **if...elseif...else statemen**t - executes different codes for more than two conditions
- **switch** statement - selects one of many blocks of code to be executed

If/Else

```php
<?php
    $hour = date("H"); //H - 24-hour format of an hour (00 to 23)
    if ($hour < 18) {
        echo "Have a good day!";
    }
    else {
        echo "Have a good evening!";
    }
?>
```

This snippet prints a greeting depending on the time of day that you run it.

Loops

In PHP, we have the following loop types::

- **while - loops** through a block of code as long as the specified condition is true
- **do...while - loops** through a block of code once, and then repeats the loop as long as the specified condition is true
- **for - loops** through a block of code a specified number of times
- **foreach - loops** through a block of code for each element in an array

While Loop Example

```php
<?php
    $x = 1;
    while($x <= 5) {
        echo "The number is: $x <br>";
        $x++;
    }
?>
```

This snippet outputs:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

For Loop Example

```php
<?php
    for ($x = 0; $x <= 5; $x++) {
        echo "The number is: $x <br>";
    }
?>
```

This snippet also outputs:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Do While Example

```
<?php
    $x = 1;
    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 0);
?>
```

In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false.

```
The number is: 1
```

For Each Example

```php
<?php
    $colors = array("red", "green", "blue", "yellow");

    foreach ($colors as $value) {
        echo "$value <br>";
    }
```

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

```
red
green
blue
yellow
```

Functions

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

User Defined Functions

A user-defined function declaration starts with the word function:

```php
<?php
    $userName = 'lou';

    function printUserName(){ //function declaration
        global $userName;
        echo $userName;
    }

   printUserName(); // function call

?>
```

lou@somewhere.com

User Defined Functions with Arguments

Arguments are specified after the function name, inside the parentheses as variables.

```php
<?php
    function printEmail($name){
        echo $name,"@somewhere.com";
    }
    printEmail("lou"); // function call
?>
```

```
lou@somewhere.com
louierusso@somewhere.com
```

User Defined Functions with Arguments

Multiple arguments are separated by commas:

```php
<?php
    function printEmail($name, $alias){
        echo $name,"@somewhere.com <br/>";
        echo $alias,"@somewhere.com";

    }
    printEmail("lou","louierusso"); // function call
?>
```

```
lou@somewhere.com
louierusso@somewhere.com
```

Functions - Returning Values

To let a function return a value, use the **`return`** statement:

```php
<?php
  function sum(int $x, int $y) {
     $z = $x + $y;
     return $z;
  }
?>
```

Arrays

An array stores multiple values in one single variable.

In PHP, the array( ) function is used to create an array:

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
  ?>
```

Array Length

The count( ) function is used to return the length (the number of elements) of an array:

```php
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo count($cars);
?>
```

Array Types

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with a numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

Indexed Arrays

The index can be assigned automatically (index always starts at 0), like this:

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

```
I like Volvo, BMW and Toyota.
```

Looping through Arrays

To loop through and print all the values of an indexed array, you could use a `for` loop, like this:

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    $arrlength = count($cars);

    for($x = 0; $x < $arrlength; $x++) {
        echo $cars[$x];
        echo "<br>";
    }
?>
```

```
Volvo
BMW
Toyota
```

Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
//option 1
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

//option 2
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Associative Arrays

The named keys can then be used in a script:

```php
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
    echo "Peter is " . $age['Peter'] . " years old.";
?>
```

```
Peter is 35 years old.
```

Looping through Associative Arrays

To loop through and print all the values of an associative array, you could use a `foreach` loop, like this:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Global and Local Scope

```php
<?php
    $userName = 'lou';

    function printGreeting(){
        $greeting = "Welcome";
        echo $greeting , " ";
    }

    function printUserName(){
        global $userName;
        echo $userName; //this is fine
    }

    printGreeting();
    printUserName();

?>
```

Online Resources

- VS Code Shortcuts
  - https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf
  - https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf
  - https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf
- PHP Tutorial Content: https://www.w3schools.com/php
-

Practice Exercises

- https://www.w3schools.com/php/exercise.asp
-