**PHP CRASH COURSE**

**Embedding PHP in HTML**

1. PHP Tags: <?php ?>

2. PHP Statements: <? echo '<p>Order processed.</p>'; ?>

3. Comments: /\*\*/; //; #

**Adding Dynamic Contenct**

1. Text
2. Function: Date()

**Accessing Form Variables**

1. ***Form Variables***

$\_POST['name']: post method

$\_GET['name']: get method

$\_REQUEST['name']: both

Output user data: echo htmlspecialchars()

1. ***String Concatenation:***

'.' --> '+'

use:

echo htmlspecialchars($tireqty).' tires<br />';

or:

$tireqty = htmlspecialchars($tireqty);

echo "$tireqty tires<br />";

1. **Variables and Literals**
2. "***double quotes***": the variable name is replaced with its value (interpolation)
3. *'****single quote****'* : the variable name is sent unaltered
4. ***heredoc*** <<<*END*  
    this is the string(can pass value)

*END;*

1. ***nowdoc*** <<<'*END*’

this is the string (can not pass value)

*END*;

1. **Variables and Literals**

var : $tireqty ( biến)

literal : ' tires<br />' (chuỗi ký tự)

**Understanding Identifiers:**

Indentifiers: names of variables

- case sensitive

- can be same name as function

\*function names: not case sensitive

**Examining Variable Types:**

1. **Variable Types**: Int,Float,String,Bool,Array,Object

**NULL** : none value , have been unset or value = NULL

**Resource**: reference to a more complex object

**Callable**: functions that are passed to other functions

1. **Type strength**: weakly typed, the type of a variable is determined by the value assigned to it

1. **Type casting***: $totalamount =* ***(float)****$totalqty;*
2. ***Variable Variables:*** enable you to change the name of variable dynamically.

$varname =‘tireqty’

**$$**varname = 5 ⬄ $tireqty = 5

**Declaring and using Constants:**

* Cannot be changed elsewhere, names of the constants appear in uppercase, names of the constants appear in uppercase

define('TIREPRICE', 100);

echo TIREPRICE;

**Understanding Variable Scope**

1. **Variable**
   1. Local
      * declared within a function is local and can only be accessed within that function
      * Can have same name in different functions
      * Local variables are deleted as soon as the function is completed
   2. Global
      * Global scope refers to any variable that is defined outside of any function
      * Global variables can be accessed from any part of the script that is not inside a function.
      * Use, acess in function:

*+ global $a, $b;*

*$b = $a + $b*

*+ $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];*

* 1. Static
     + You want a local variable to not be deleted.
     + Variable will still have the information it contained from the last time.
     + **Note**: The variable is still local to the function.
     + Use:

+ static $rememberMe;

* 1. Function parameters:
     + function myTest($para1,$para2,...) {

// function code

}

* 1. Supeprglobal:
     + **$GLOBALS** — An array of all global variables
     + **$\_SERVER** — An array of server environment variables
     + **$\_GET** — GET method
     + **$\_POST** — POST method
     + **$\_COOKIE** — An array of cookie variables
     + **$\_FILES** — An array of variables related to file upload
     + **$\_ENV** — An array of environment variables
     + **$\_REQUEST** — An array of all user input including the contents of input including $\_GET, $\_POST, and $\_COOKIEs
     + **$\_SESSION** — An array of session variables

1. **Scope rule:**

* ***Superglobal*** variable are visible everywhere in the script
* *Constant* visible ***globally***, use inside & outside function
* Global variables declared in a script are visible throughout that script, but not insidefunctions.
* Variables inside functions that are declared as global refer to the global variables of the  
  same name
* Variables created inside functions and declared as static are invisible from outside the  
  function but keep their value between one execution of the function and the next
* Variables created inside functions are local to the function and cease to exist when the  
  function terminates

**Using Operators**

**1. Arithmetic Operators:** +,-,\*,/,%

**2. String Operators:** .

**3. Assignment Operators:** ++,--,+=,…

- Reference Operator

*$a = 5;  
$b = &$a;  
$a = 7;* // $a and $b are now both 7

+ Both $a and $b point to the same piece of memory. You can change this by unsetting one of them as follows: unset($a)

+ Unsetting does not change the value of $b (7) but does break the link between $a and the  
value 7 stored in memory

**4. Comparison Operators**

**5. Logical Operators**

**6. Bitwise Operators**

**7. Other Operators**

- **Ternary Operator** :

*condition ? value if true : value if false*

- **Error Suppression Operator**:

*$a = @(57/0)*

- **The Execution Operator**:

*$out = `dir c:`;  
echo '<pre>'.$out.'</pre>';*

- **Array Operators**:

*‘$a +$b’ Returns an array containing everything in $a and $b*

*-* **The Type Operator**: check whether an object is an instance of a particular class, use **instanceof:**

class sampleClass{};  
$myObject = new sampleClass();  
if ($myObject **instanceof** sampleClass)  
echo "myObject is an instance of sampleClass";

**Understanding Precedence and Associativity**

**Using Variable Handling Functions**

**1. Using Variable Handling Functions**

- gettype(), settype():

$a = 56;  
 echo gettype($a).'<br />';  
 settype($a, 'float');  
 echo gettype($a).'<br />';

- true or false:

* is\_array()
* is\_double(), is\_float(), is\_real()
* is\_long(), is\_int(), is\_integer()
* is\_string()
* is\_bool()
* is\_object()
* is\_resource()
* is\_null()
* is\_scalar()—Checks whether the variable is a scalar—that is, an integer, boolean,string, or float
* is\_numeric()—Checks whether the variable is any kind of number or a numeric string
* is\_callable()—Checks whether the variable is the name of a valid function

**2. Testing Variable Status**

- bool isset(mixed *var[,* mixed var[,...]])  
 - void unset(mixed var[, mixed var[,...]])

- bool empty(mixed var

**3. Reinterpreting Variables**

- Get the integer,float,str value of a variable

* + - echo intval($a).'</br>';
    - echo floatval($a).'</br>';
    - echo strval($a).'</br>';

**Making Decisions with Conditionals**

Control structures: **conditional** structure(or branching) and repetition structures (or **loops**).

**Repeating Actions Through Iteration**

**Breaking Out of a Control**

**Alternative Control Structure**

- **endif**, **endswitch**, **endwhile**, **endfor**, or **endforeach**, depending on which control structure is being used.

**if** ($totalqty == 0) :  
 echo "You did not order anything on the previous page!<br />";  
 exit;  
end**if**;