

# PHP Language

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- `date()` function
- Passing variables with `get` and `post`
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# This lecture covers

- **tags**
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- `date()` function
- Passing variables with `get` and `post`
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# PHP tags

`<?php`

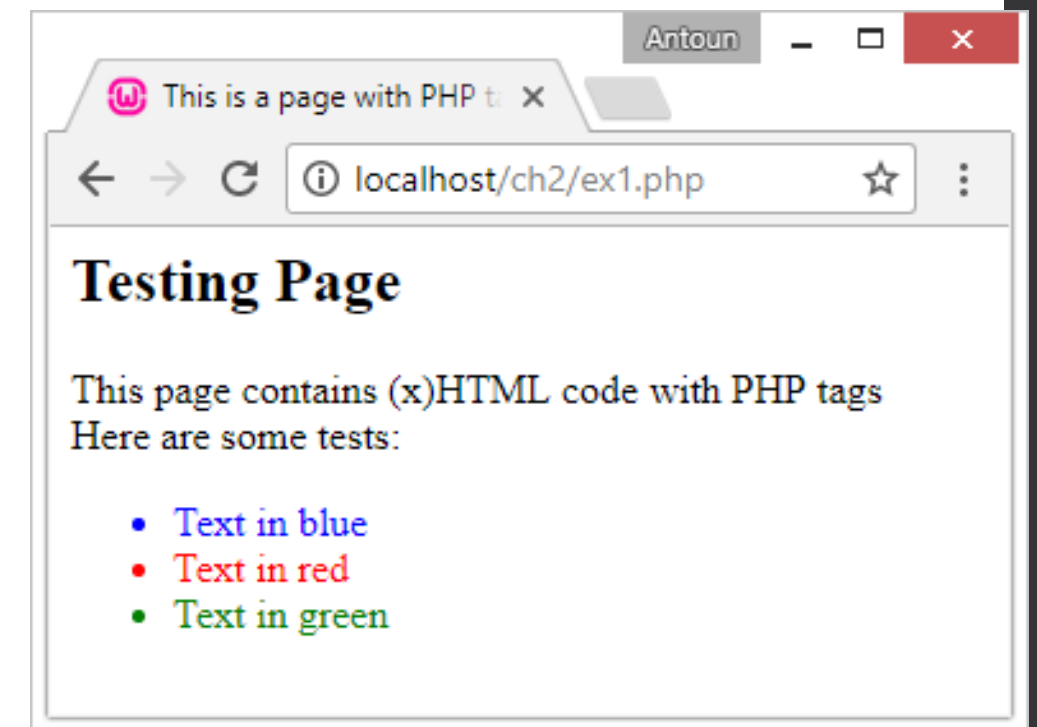
`//PHP code is here`

`?>`

- Other tags to use PHP
  - `<? ?>`, `<% %>`, etc...

# PHP tags

```
1 <html>
2   <head>
3     <title>This is a page with PHP tags</title>
4     <style type="text/css">
5       .blue{color:blue;}
6       .red{color:red;}
7       .green{color:green;}
8     </style>
9   </head>
10  <body>
11    <h2>Testing Page</h2>
12    <p>This page contains (x)HTML code with PHP tags</br>
13    <?php
14      //here we put PHP code
15    ?>
16    Here are some tests:</p>
17    <ul>
18      <li class="blue">Text in blue</li>
19      <li class="red">Text in red</li>
20      <li class="green">Text in green</li>
21    </ul>
22    <?php
23      //more PHP code
24    ?>
25  </body>
26</html>
```



# This lecture covers

- tags
- **display text**
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Display text

```
1 <html>
2   <head>
3     <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
4     <title>echo instruction</title>
5   </head>
6   <body>
7     <h2>Display text with PHP</h2>
8     <p>This sentence was written in x(HTML)</br>
9     <?php
10      echo "however, this sentence is written in PHP.";
11    ?>
12   </p>
13 </body>
14 </html>
```



# Display text

```
echo "however, this sentence is written in PHP.";
```

- echo
  - Display instruction
  - between " "
  - then;



# Display text

```
<?php
    echo "this sentence is written <strong>only</strong> in PHP";
?>
```

- "only" will be written in bold due to the tags <strong> and </strong>
- How to display a double quotation?

```
<?php
    echo "this sentence is written \"only\" in PHP";
?>
```

# This lecture covers

- tags
- display text
- **comments**
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Comments

- Just for you i.e. coder
- Help you recheck your PHP code

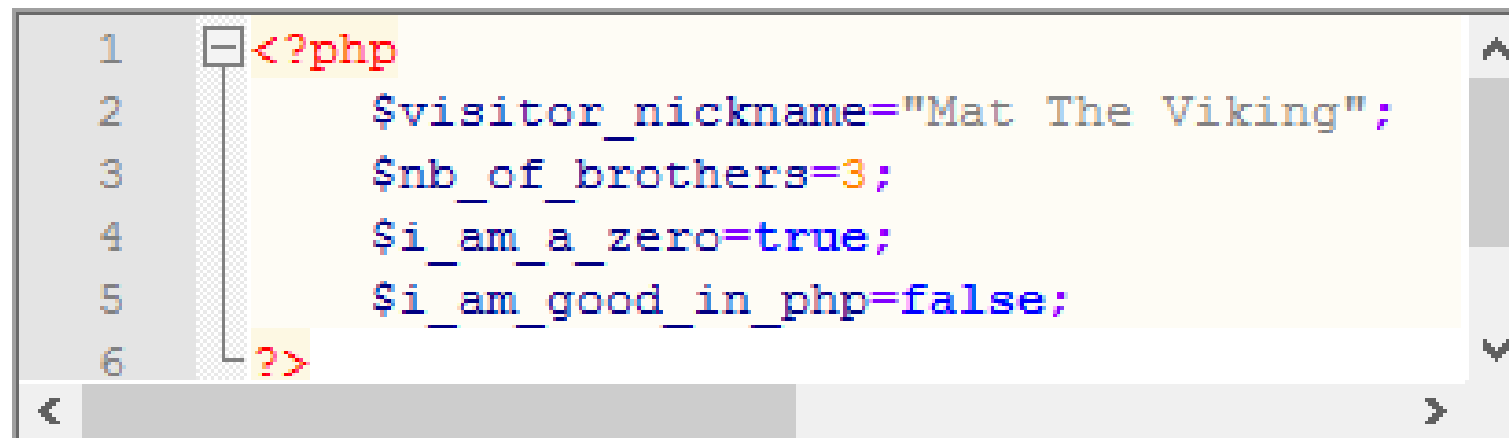
```
1  <?php
2      echo "I live in Beirut"; // This line indicates where I live
3      //the following line indicates my age
4      echo "I am 92 years old";
5      //End of line comment
6
7      /*
8      comments
9      spanning
10     over multiple
11     lines
12     */
13     # End of line comment as in Shell
14  ?>
```

# This lecture covers

- tags
- display text
- comments
- **variables**
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- `date()` function
- Passing variables with `get` and `post`
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Variables

- `$name_variable`
- Assignment
- 3 sorts of " data" in a variable : text, numbers, or boolean.
  - text between double quotation
  - numbers and boolean without quotation



```
1 <?php
2     $visitor_nickname="Mat The Viking";
3     $nb_of_brothers=3;
4     $i_am_a_zero=true;
5     $i_am_good_in_php=false;
6 ?>
```

The screenshot shows a code editor with a light yellow background. On the left, there is a vertical line with numbers 1 through 6. The code is written in a monospaced font with syntax highlighting: opening and closing PHP tags are red, variable names are blue, string literals are green, and numeric and boolean literals are orange. The code assigns values to four variables: \$visitor\_nickname, \$nb\_of\_brothers, \$i\_am\_a\_zero, and \$i\_am\_good\_in\_php. The editor has a scrollbar on the right and a status bar at the bottom.

# Assignment and display



The image shows a code editor window with the following PHP code:

```
1 <?php
2     $visitor_nickname="Mat The Viking";
3     echo $visitor_nickname;
4     echo "$visitor_nickname";
5     echo "Hello $visitor_nickname";
6 ?>
```

Below the code editor is a web browser window titled "Antoun" showing the output of the PHP script at the URL "localhost/ch2/ex5.php". The output is displayed on three separate lines:

```
Mat The Viking
Mat The Viking
Hello Mat The Viking
```

*! Code was altered to  
Display each echo on a  
separate line*

# Assignment and display

```
1  <?php
2      $number = 2 + 4; //6
3      $number = 5 - 1; //4
4      $number = 3 * 5; //15
5      $number = 9 / 2; //4.5
6      $number = 3 * 5 + 1; //16
7      $number = (1 + 2 ) * 2; //6
8      $number = ($number + 5) * $number; //6
9  ?>
```

# Variables, types and operators

- implicit variables typing in PHP
- no need to declare their type beforehand
- nor even to initialize them before their use
- identifiers preceded by `$` (example `$toto`)
- type
  - integer
  - double
  - string
  - array
  - object
  - boolean
- convert a variable into a primitive type by casting (as in C)
  - `$str = "12"; // $str is the string "12";`
  - `$nbr = (int)$str; // $nbr worth the number 12`



# Variables, types and operators

- some functions :
  - `empty($var)` : returns true if the variable is empty
  - `isset($var)` : returns true if the variable exists
  - `unset($var)` : destroys a variable
  - `gettype($var)` : returns the type of the variable
  - `settype($var, "type")` : converts the variable to type type (cast)
  - `is_long()`, `is_double()`, `is_string()`, `is_array()`, `is_object()`, `is_bool()`, `is_float()`, `is_numeric()`, `is_integer()`, `is_int()`...
- a variable may have as its identifier the value of another variable
  - syntax : `${$var} = value;`
- example :
  - `$toto = "foobar";`
  - `${$toto} = 2002;`
  - `echo $foobar; // 2002`

# Variables, types and operators

- range limited to block
- arithmetic operators :
  - + (addition), - (subtraction), \* (multiplication), / (division), % (modulo), ++ (increment), -- (decrement) pre or post fixed
- Assignment operators :
  - = (assignment), \*= ( $\$x*=\$y$  equivalent to  $\$x=\$x*\$y$ ), /=, +=, -=, %=
- logical operators :
  - and, && (and), or, || (or), xor (exclusive or), ! (no)
- comparison operators :
  - == (equality), < (strict lower), <= (lower), >, >=, != (difference)
- ternary operator :
  - (condition)?(expression1):(expression2);
  - `$toto=11;`  
`$nbr = ($toto<10)?($toto):($toto%10); // $nbr = 1`

# This lecture covers

- tags
- display text
- comments
- variables
- **constants**
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- `date()` function
- Passing variables with `get` and `post`
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

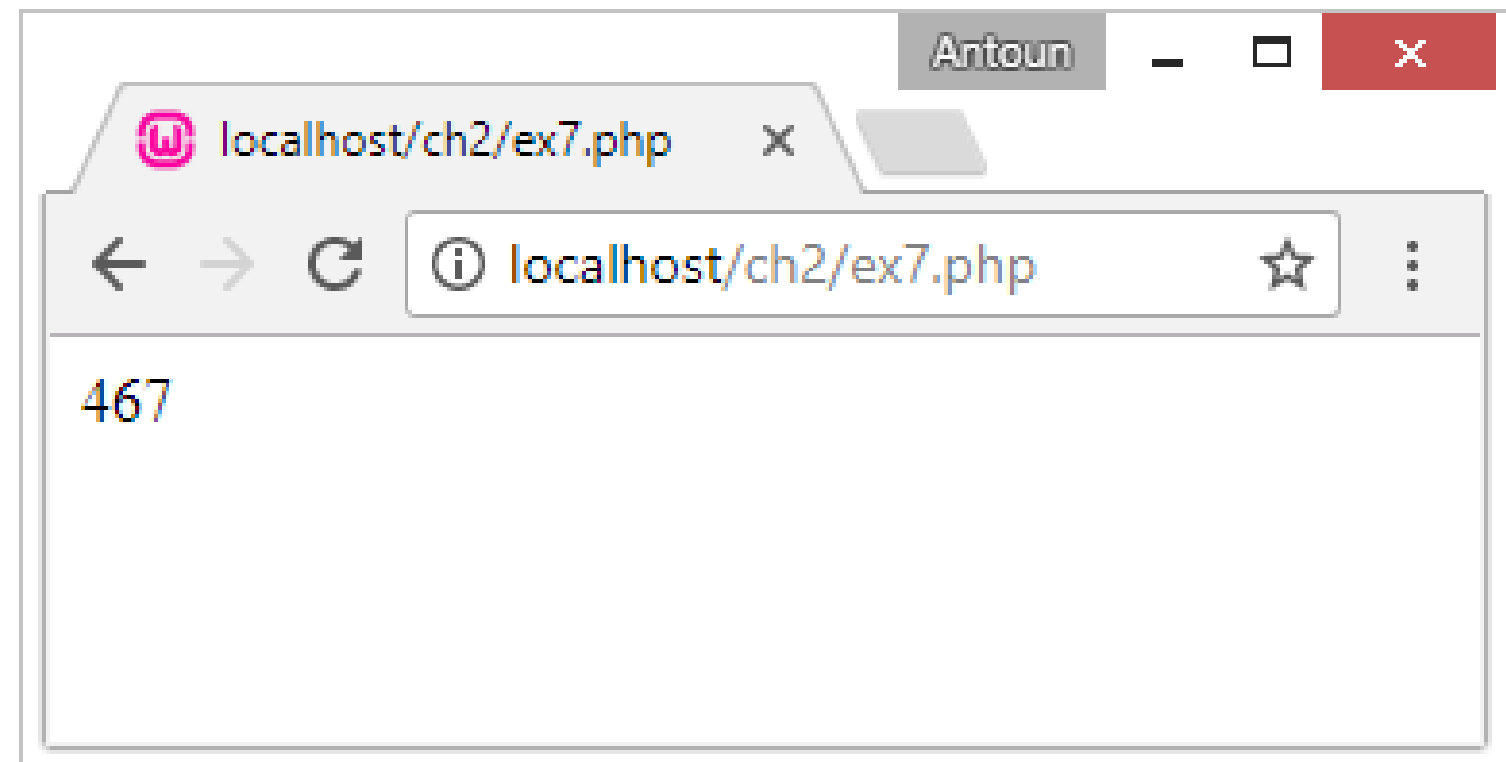
# Constants

- value
  - fixed once and for all
  - not modifiable
- identifier
  - no \$ symbol
  - insensitive to case
- `define ("var", value);`

```
<?php
    define ("author", "Foobar");
    echo author; // Foobar
    define (MY_YEAR, 1980);
    echo mY_Year; // 1980
?>
```

# Booleans

```
1 <?php
2     if(0)
3         echo 1; //false
4     if("")
5         echo 2; //false
6     if("0")
7         echo 3; //false
8     if("00")
9         echo 4;
10    if('0')
11        echo 5; //false
12    if('00')
13        echo 6;
14    if(" ")
15        echo 7;
16 ?>
```

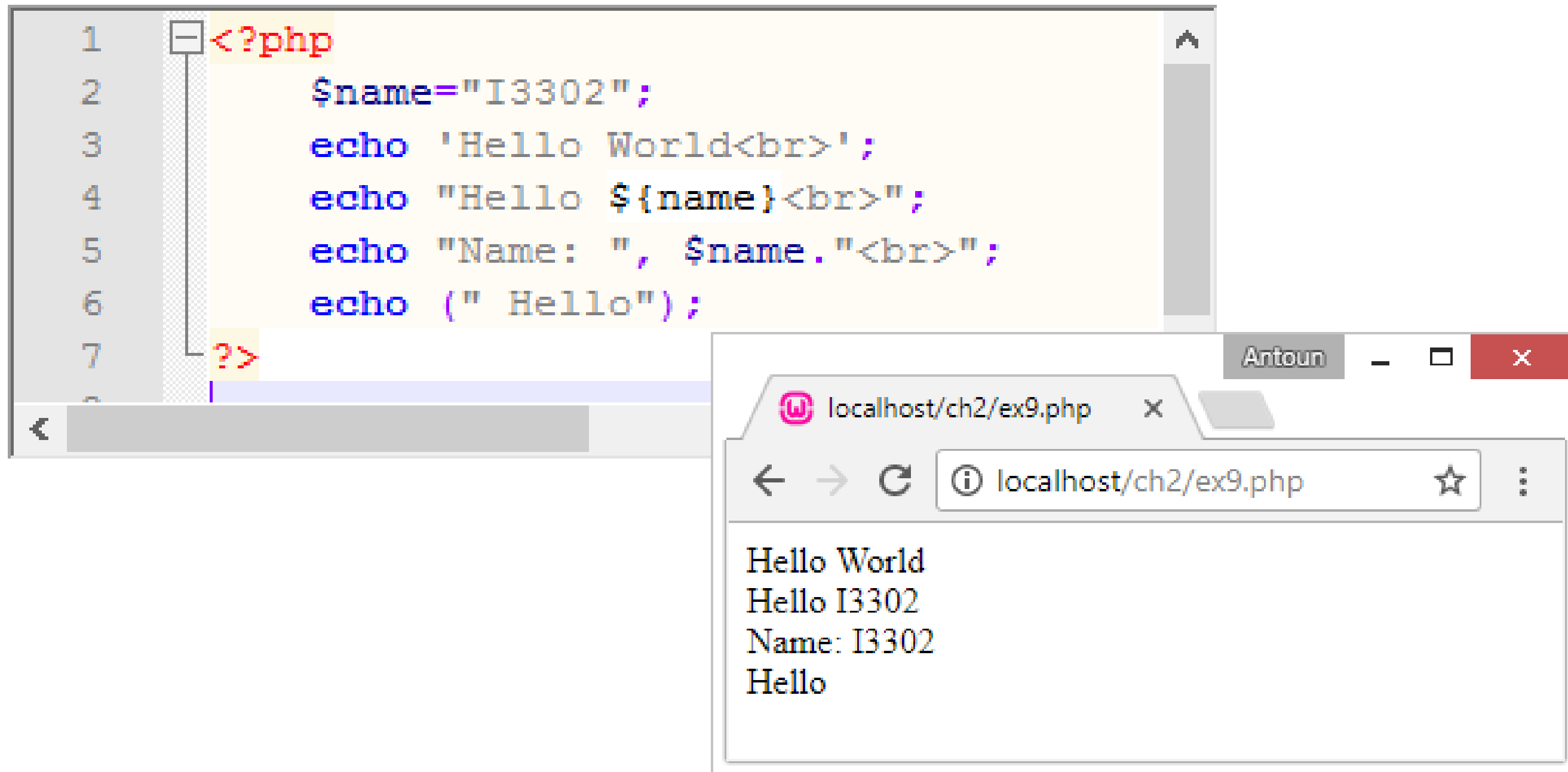


# Strings and Concatenation

- not limited in number of characters
- delimited by
  - ' ' → allow the evaluation of variables and special characters
  - " " → allow the evaluation of variables and special characters
- \n (new line) LF (Line Feed)
- \r (return to a new line) CR (carriage return)
- \t (horizontal tab)
- \\ (backslash)
- \\$ (\$ character)
- \" (double quote)

```
1 <?php
2     $first_name="Donald";
3     $last_name='Trump';
4
5     echo "Name: $first_name"; // Name: Donald
6     echo 'Name: $first_name'; // Name: $first_name
7     echo "Hello World !\n";
8
9     $foo="Hello";
10    $bar="World";
11    echo $foo.bar; // HelloWorld
12    echo $foo . bar; // HelloWorld
13    echo $foo ." ". bar; // Hello World
14
15    $name="Henry";
16    echo $name."IV"; //HenryIV
17    $whoiam=$name."IV";
18    echo $whoiam; //HenryIV
19
20    $out ='Here';
21    $out.=" and there ...";
22    echo $out; // Here and there ...
23
24
```

# Strings and echo



The image shows a code editor on the left and a web browser on the right. The code editor contains a PHP script with the following lines:

```
1 <?php
2     $name="I3302";
3     echo 'Hello World<br>';
4     echo "Hello ${name}<br>";
5     echo "Name: ", $name."<br>";
6     echo (" Hello");
7 ?>
```

The web browser on the right shows the output of the script at the URL `localhost/ch2/ex9.php`. The output is:

```
Hello World
Hello I3302
Name: I3302
Hello
```

# Functions for strings

- `strlen($str)`
  - returns the number of characters in a string
- `strtolower($str)`
  - returns str in lowercase
- `strtoupper($str)`
  - returns str in uppercase
- `trim($str)`
  - Removal of start and end spaces
- `substr($str,$i,$j)`
  - returns a sub-string of \$str of size \$j and starting at position \$i
- `strcmp($str1,$str2)`
  - comparaison of 2 strings
- `strpos($str , $val [, $offset] )`
  - looks for the numeric position of the first occurrence of \$val in \$str, starts from \$offset
- `addslashes($str)`
  - despecializes special characters
- `ord($char)`
  - returns the ASCII value of the \$char



# Functions and References

```
1  <?php
2      $foo=100;
3      $foobar=&$foo;
4      $foo++;
5      echo $foobar; // 101
6
7      function change($var)
8      {
9          $var++;
10     }
11     $nbr=1;
12     change(&$nbr);
13     echo $nbr; //2
14  ?>
```

# Mathematical functions

- `abs($x)`
  - absolute value
- `ceil($x)`
  - rounded upper
- `floor($x)`
  - lower roundness
- `pow($x,$y)`
  - x exponent y
- `round($x,$i)`
  - x rounded to the ith decimal
- `max($a, $b, $c ...)`
  - returns the maximum value argument
- `pi()`
  - returns the value of Pi
- ...
- `cos, sin, tan, exp, log, min, pi, sqrt...`
- `M_PI`
  - value of pi
  - 3.14159265358979323846
- `M_E`
  - value of e
  - 2.7182818284590452354

# Random numbers

- `rand([$x[, $y]])` :  
random integer value between
  - 0 and `RAND_MAX` if `x` and `y` are not defined
  - `x` and `RAND_MAX` if only `x` is defined
  - `x` and `y` if these two parameters are defined
- `srand()`
  - initialization of the random generator
- `getrandmax()`
  - returns the value of the largest integer that can be generated
- algorithm used by `rand()` is slow and predictable
- `mt_rand()`
  - faster and safer
  - is based on cryptography
- `mt_rand()` goes with
  - `mt_rand([$x[, $y]])`
  - `mt_srand()`
  - `mt_getrandmax()`

# Numbers formatting

- `number_format ($nbr[, $dec, [$a, $b]])`
  - returns a string representing the number \$nbr
  - \$dec decimals after formatting
  - \$a serving as a comma
  - \$b the separator of thousands
- by default
- `$a = "."`
- `$b = ","`

```
number_format (1000000.3333) ;//1,000,000
number_format (1000000.3333, 2) ;//1,000,000.33
number_format (1000000.3333, 2, ",", ".", "-") ;//1.000.000,33
```

# Hint

- Both codes below give exactly the same result:

```
1  <?php
2      if($var==23)
3      {
4          echo "Some text";
5      }
6  ?>
7
8  <?php
9      if($var==23)
10     {
11     ?>
12         Some text
13     <?php
14     }
15     ?>
```

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- **control instructions**
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Switch

```
1  <?php
2      switch ($grade)
3      {
4          case 0:
5          case 1:
6          case 2:
7          case 3:
8          case 4: echo "You are nothing !"; break;
9          case 5:
10         case 6:
11         default: echo "great";
12     }
13  ?>
```

# Iteration

```
for( ... ; ... ; ... ) { ... }
```

```
while( ... ) { ... }
```

```
do { ... } while( ... );
```



# Break and continue

- break

- Quit a loop

```
while($nbr = $tab[$i++]) {  
    echo $nbr."<br />";  
    if($nbr == $stop)  
        break;  
}
```

- continue

- Skip remaining instructions and loop

```
for($i=1; $i<=10; $i++) {  
    if($tab[$i] == $val)  
        continue;  
    echo $tab[$i];  
}
```

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- **stop a script**
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Stopping a whole script

- To stop a script before its normal end, use one of the following 2 functions
- **die**
  - Stops the script and displays an error message in the browser  
`if(mysql_query($query) == false)`  
`die("Error in the dabase with query : <br />$query");`
- **exit**
  - Stops the script without displaying an error message  
`function foobar() {exit();}`

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- **arrays**
- Associative arrays
- functions
- `date()` function
- Passing variables with `get` and `post`
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Arrays

- An array/table is of type array
- Accepts elements of any type and can be of different types
- Array can be initialized with the syntax array

```
$tab_colors = array('red', 'yellow', 'blue', 'white');  
$tab = array('foobar', 2002, 20.5, $name);
```
- Arrays can be initialized on the go.

```
$names[ ] = "Clément";      $region[0] = "Paris";  
$names[ ] = "Justin";      $region[1] = "Londres";  
$names[ ] = "Tanguy";      $region[2] = "Lisbonne";
```
- Calling an element in an array is done via its index (origin is 0 as in C)

```
echo $tab[10];              // to access the 11th element
```

# Arrays

- Array traversal

```
$tab = array('Hugo', 'Jean', 'Mario');
```

- example 1 :

```
$i=0;
while($i < count($tab)) {
    echo $tab[$i].'\n';
    $i++;
}
```

- example 2 :

```
foreach($tab as $elem) {
    echo $elem."\n";
}
```

// \$elem takes successively all the values of the array \$tab

# Arrays

- To modify the elements after the foreach loop
  - Use the reference
  - Break the link after the loop

- example

```
foreach($tab as &$elem) {  
    $elem++;  
}  
unset($elem);
```

- *explication: otherwise \$elem will be always linked to the last element of the array even after the loops ends*

# Functions for arrays

- `count($tab), sizeof`
  - Return the number of elements in an array
- `in_array($var,$tab)`
  - Says whether the value of \$var exists in the array \$tab
- `list($var1,$var2...)`
  - Transforms a list of variables into an array
- `range($i,$j)`
  - Returns an array containing an interval of values
- `sort($tab)`
  - Sorts alphanumerically the elements in the array
- `rsort($tab)`
  - Sorts in reverse order the elements in the array
- `implode($str,$tab), join`
  - Return a string containing the elements of the array \$tab joined to the string \$str
- `explode($delim,$str)`
  - breaks a string \$str into an array
- `array_merge($tab1,$tab2,$tab3...)`
  - Concatenate arrays passed in arguments
- `array_rand($tab)`
  - Returns randomly an element from the array
- `shuffle($tab)`
  - Shuffles the elements in the array



# Arrays

- Array variables are not evaluated in a middle of a string delimited by ""
- example :
  - `echo "$tab[3]";` // invalid syntax
  - `echo $tab[3];` // valid syntax
  - `echo "${tab[3]}";` // valid syntax

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- **Associative arrays**
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# Associative arrays

- Associative array or dictionary or hashtable
- Associate to each element a key; the key is of type string
- Initialization of an associative array is similar to that of a normal array  
`$person = array("LastName" => "Cesar", "FirstName" => "Jules");`  
OR  
`$person["LastName"] = "Cesar";`  
`$person["FirstName"] = "Jules";`
- Here, to the key `LastName`, we associate the value `César`

# Associative arrays

- traversal

```
$person = array("LastName" => "Cesar", "FirstName" => "Jules");
```

```
foreach($person as $elem)  
{ echo $elem; }
```

- Access directly the elements without passing by the keys

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

- Access to the keys also

# Associative arrays functions

- `array_count_values($tab)`
  - Returns an array containing the values of the array `$tab` as keys and their frequencies as values (useful to evaluate redundancies)
- `array_keys($tab)`
  - Returns an array containing the keys of `$tab`
- `array_values($tab)`
  - Returns an array containing the values of `$tab`
- `array_search($val,$tab)`
  - Returns the key associated to the value `$val`
- An element in an array can be another array
- Associative arrays allows to preserve a data structure

# Sorting Arrays

- `sort()`, by value, dropping the keys (ex. `$arr[i]` can be changed to another value)
- `asort()`, by value, maintaining the keys
- `ksort()`, by key, maintaining the values
- `rsort()`, `sort()` in reverse order
- `arsort()`, `asort()` in reverse order
- `krsort()`, `ksort()` in reverse order

# Superglobal Arrays

- `$_GET`
- `$_POST`
- `$_REQUEST`
- `$_SERVER`
- `$_SESSION`
- `$_COOKIE`

```
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
```

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>

<html>
<body>
```

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
    Name: <input type="text" name="fname">
    <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
```

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
```

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- **functions**
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants



# Functions in PHP

- Arguments: No need to specify any type
- Return a value: optional
- Call: no need to respect its prototype (number of parameters)
- Function identifier: case insensitive

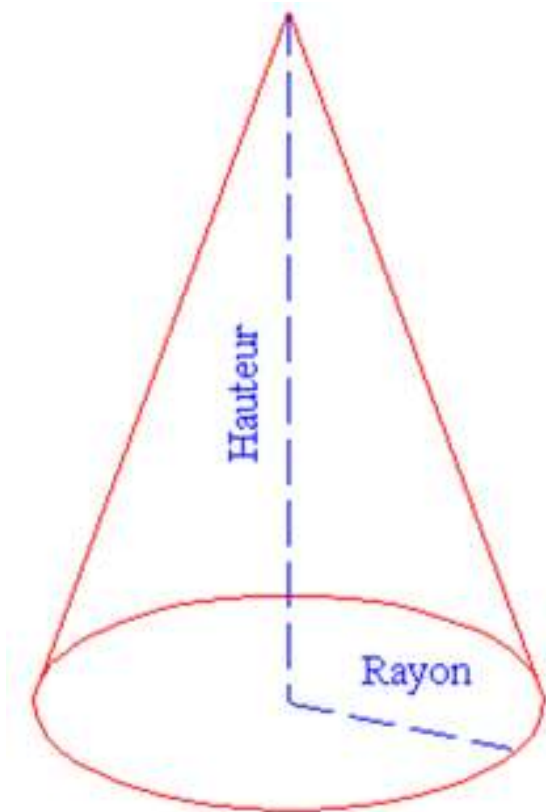
```
function myfunction($toto)
{
    $toto += 15;
    echo "Hello !";
    return ($toto+10);
}
$nbr = MyFunction(15.1); //40.1
```

# Functions in PHP

- Volume is :
  - $v = r * r * 3.14 * h * (1/3)$

```
<?php
function VolumeCone($radius, $height)
{
    return $radius * $radius * 3.14 * $height * (1/3);
}

$volume = VolumeCone(3, 1);
echo " a cone with radius 3 and height 1";
echo " has a volume of $volume";
?>
```



# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- **date() function**
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

# date() function

- Respects uppercase/lowercase
- Takes lot of parameters (around 30)

Parameter	Description
H	Hour
i	Minute
d	Day
m	Month
Y	Year

# Examples

- Display the year

```
<?php
    $year = date("Y");
    echo "$year";
?>
```

- Complete date + hour

```
<?php
    $day = date("d");
    $month = date("m");
    $year = date("Y");
    $hour = date("H");
    $minute = date("i");
    echo « Hello ! Today is $day/$month/$year $hour h $minute.";
?>
```

- The time is the server time

# date() function [a part]

	Description	Example returned values
d	Day of the month, 2 digits with leading zeros	01 to 31
D	A textual representation of a day, three letters	Mon through Sun
j	Day of the month without leading zeros	1 to 31
l	A full textual representation of the day of the week	Sunday through Saturday
N	ISO-8601 numeric representation of the day of week	1 (for Monday) through 7 (for Sunday)
S	English ordinal suffix for the day of the month 2 chars	st, nd, rd or th. Works well with j
w	Numeric representation of the day of the week	0 (for Sunday) through 6 (for Saturday)
z	The day of the year (starting from 0)	0 through 365
W	ISO-8601 week number of year, weeks start Monday	Example: 42 (the 42nd week in the year)
F	full textual representation of a month, January ...	January through December
m	Numeric representation of a month, with leading zeros	01 through 12
M	A short textual representation of a month, three letters	Jan through Dec
n	Numeric representation of month, without zeros	1 through 12
t	Number of days in the given month	28 through 31
L	Whether it's a leap year	1 if it is a leap year, 0 otherwise.
o	ISO-8601 year number. (added in PHP 5.1.0) ...	Examples: 1999 or 2003
Y	A full numeric representation of a year, 4 digits	Examples: 1999 or 2003
y	A two digit representation of a year	Examples: 99 or 03

# Functions

- **global**

- Modifies the scope of local variables of a function
- Associative array `$GLOBALS` allows access to global variables of a script
- `$GLOBALS["var"]` accesses variable `$var`

```
function change() {  
    global $var; // defines $var as global  
    $GLOBALS["toto"]++; // increments the global variable $toto  
    $var++; // this will affect the rest of the script  
}
```

- **static**

- Conserves the value of a local variable of a function

```
function change() {  
    static $var; // defines $var as static  
    $var++; // its value will be preserved to its second call  
}
```

# Functions

- Give a default value to arguments when declaring functions

```
function Set_Color($color="black") {  
    global $car;  
    $car["color"] = $color;  
}
```

- Force passing parameters by reference

```
function change(&$var) { // forces passing parameters by reference  
    $var += 100;  
}  
$toto = 12; // $toto is equal to 12  
change($toto); // passing by value but the function takes it as a reference  
echo $toto; // $toto is equal to 112
```



# Functions

- Since PHP4, a function can be defined after being called because compilation comes before execution
- Before: PHP3 interpreted

```
function foo()
{
    echo "Foo...";
}
foo();
bar();
function bar()
{echo "bar!<br />"; }
```
- example will display: Foo...bar!

# Functions

- To return multiple values in an array, use `list()` → parameters = variables values
- Assign to `list()` the return of a function

```
function trigo($nbr) {  
    return array(sin($nbr), cos($nbr), tan($nbr));  
}  
$r = 12;  
list($a, $b, $c) = trigo($r);  
echo "sin($r)=$a, cos($r)=$b, tan($r)=$c";
```
- displays  
sin(12)=-0,5365729180, cos(12)=0,8438539587, tan(12)=-0,6358599286

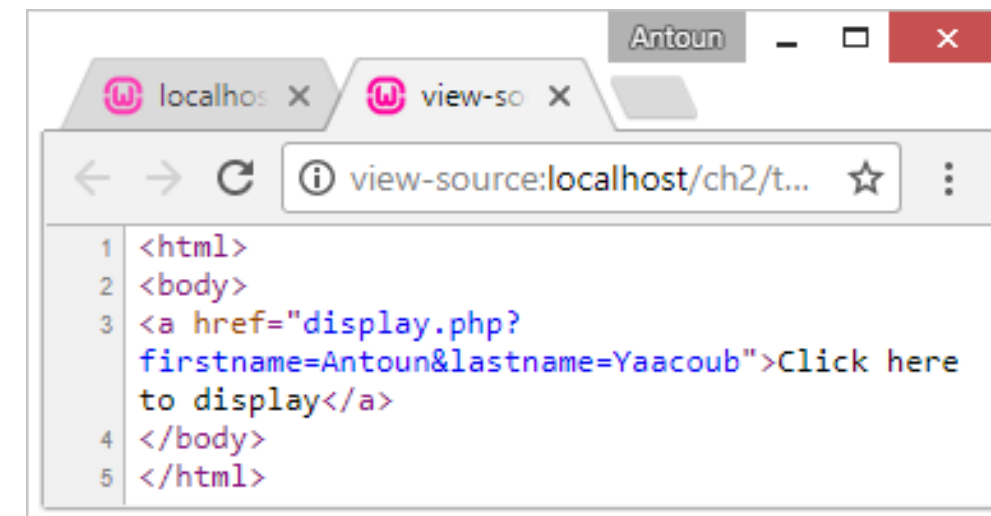
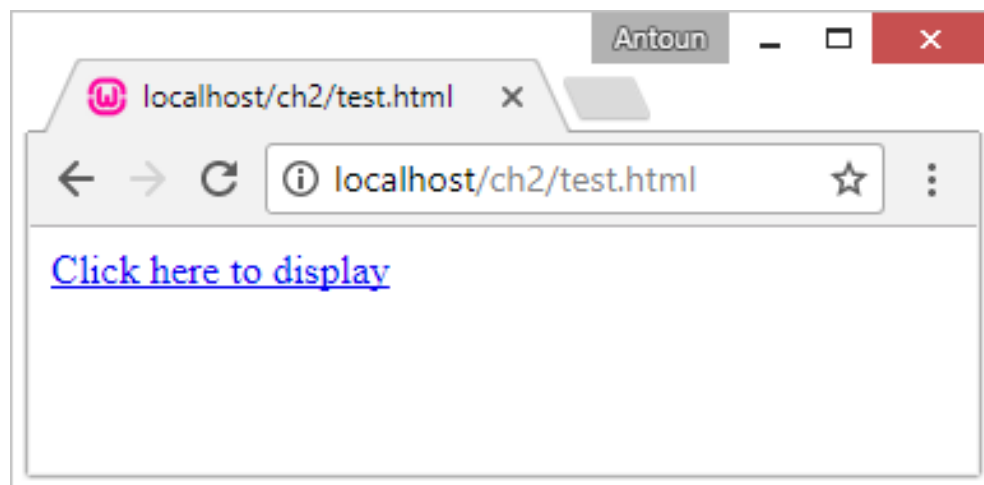
# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- **Passing variables with get and post**
- Include external files
- Dynamic functions
- Environment variables
- Environment constants

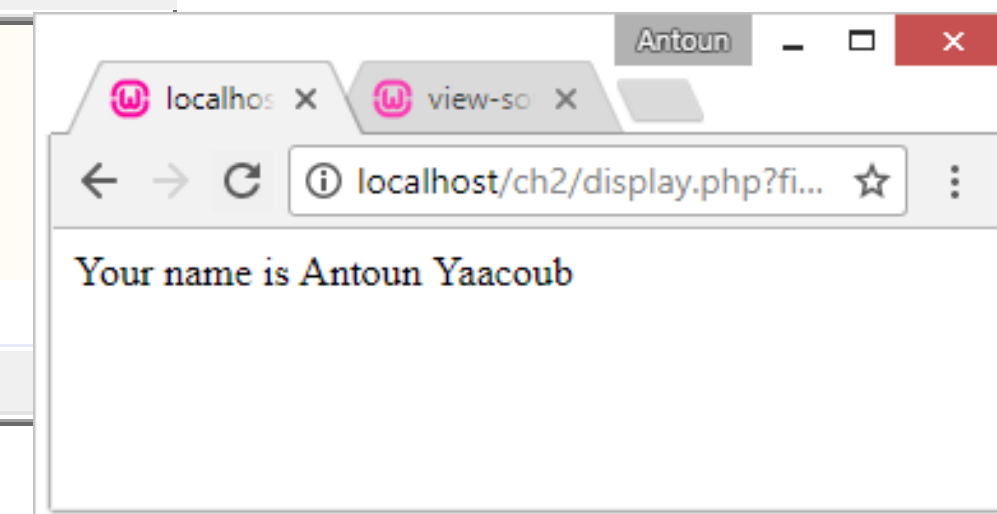
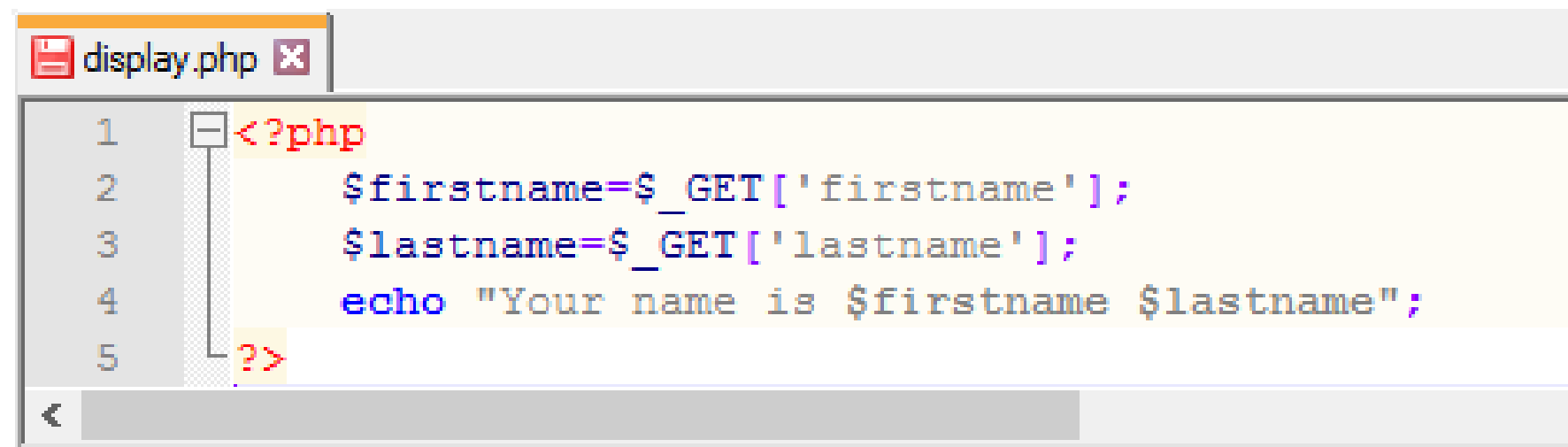
# Variables Transmission

- POST method
  - `$_POST['xxxx']`
- GET method
  - `$_GET['xxxx']`

# Variables Transmission- GET

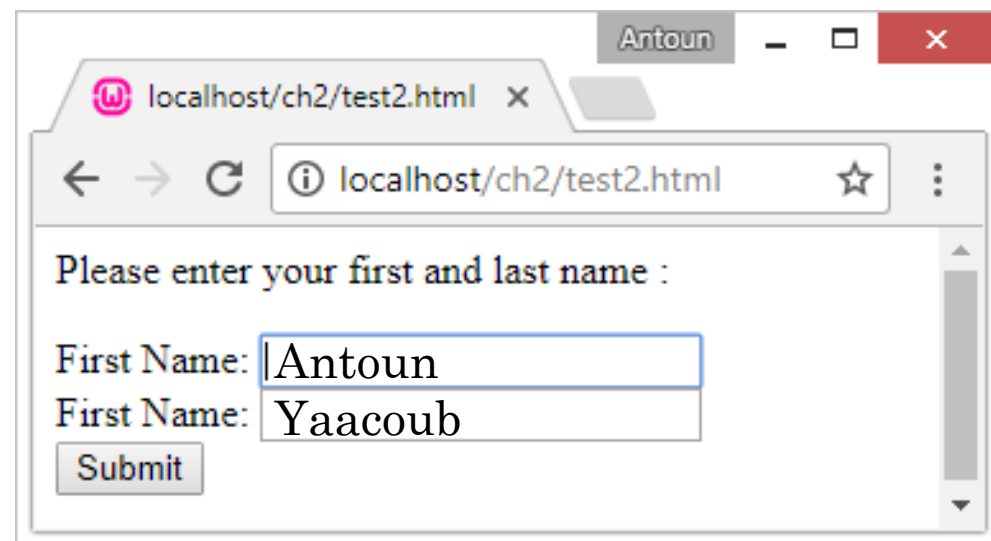


- Separate variables with & → &amp;



# Variables Transmission- POST

- Via a form



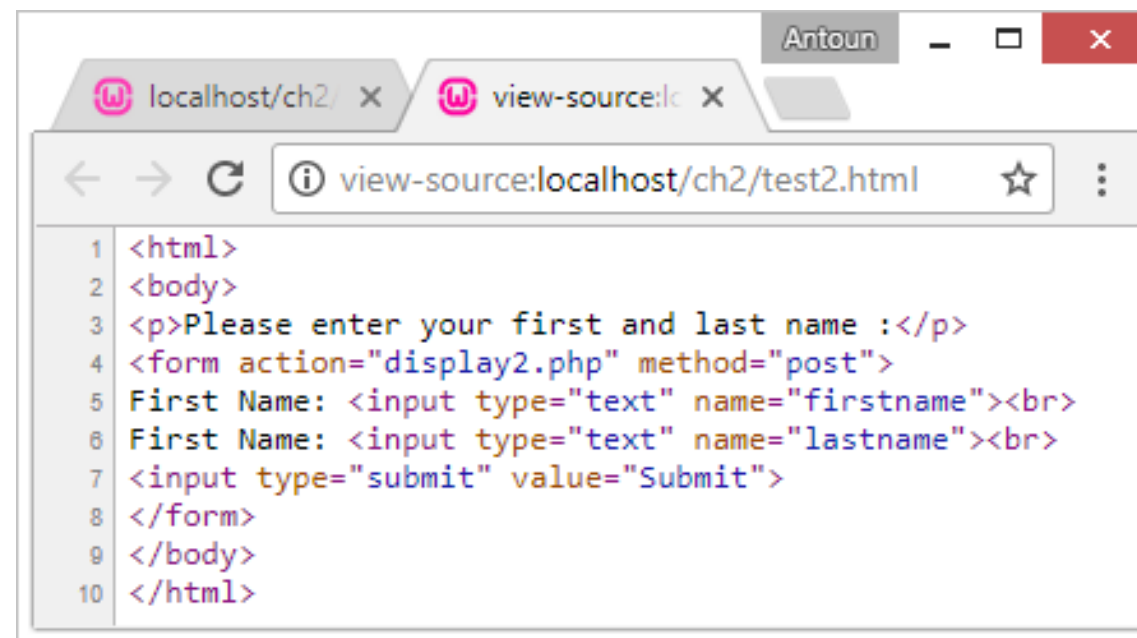
Antoun

localhost/ch2/test2.html

Please enter your first and last name :

First Name:

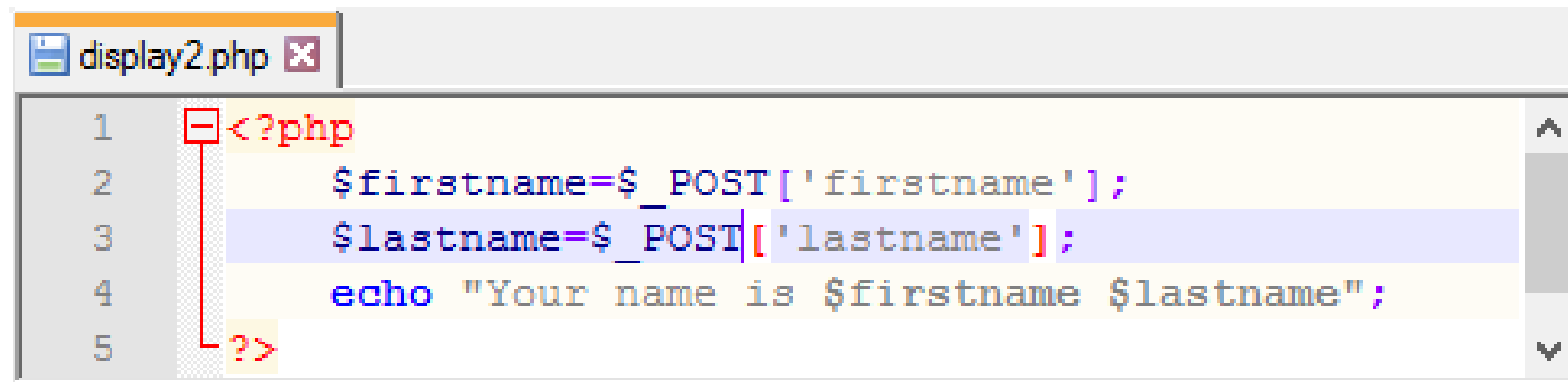
First Name:



Antoun

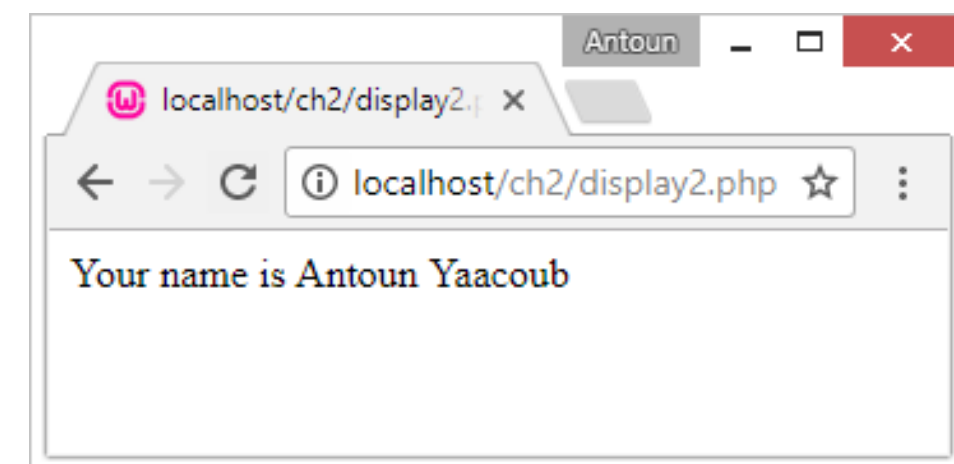
localhost/ch2/ view-source:localhost/ch2/test2.html

```
1 <html>
2 <body>
3 <p>Please enter your first and last name :</p>
4 <form action="display2.php" method="post">
5 First Name: <input type="text" name="firstname"><br>
6 First Name: <input type="text" name="lastname"><br>
7 <input type="submit" value="Submit">
8 </form>
9 </body>
10 </html>
```



display2.php

```
1 <?php
2 $firstname=$_POST['firstname'];
3 $lastname=$_POST['lastname'];
4 echo "Your name is $firstname $lastname";
5 ?>
```



Antoun

localhost/ch2/display2.php

Your name is Antoun Yaacoub

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- **Include external files**
- Dynamic functions
- Environment variables
- Environment constants

# Including external files

- Include in a PHP script the content of another file
- `require`
  - Insert the content of the file even if it's not a PHP script
  - Equivalent to `#include` in C
  - `require("file.php");`
- `include`
  - **Evaluates** and inserts at each call (even in a loop) the content of the file
  - `include("file.php");`



# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- **Dynamic functions**
- Environment variables
- Environment constants

# Dynamic Functions

- It is possible to dynamically create functions
- Assign to a string variable the name of the function to duplicate
- Pass in argument to this variable the parameters of the first function

```
function diverse($toto) { echo $toto; }  
$myfunction = "diverse";  
$myfunction("Hello !");  
// displays 'Hello !' by calling diverse()
```

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- **Environment variables**
- Environment constants

phpinfo()

x

Antoun


-

x

localhost/phpinfo.php

☆

PHP Version 5.6.31



System	Windows NT HOMEPC 6.2 build 9200 (Windows 8 Enterprise Edition) AMD64
Build Date	Jul 5 2017 22:19:43
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	D:\PHP\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API20131226,TS,VC11
PHP Extension Build	API20131226,TS,VC11
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled

# Environment variables

- PHP has a multitude of environment variables
- `phpinfo()` displays these variables, and displays the Apache server configuration
- variables → useful information [and essential]
  - `$PHP_SELF` : name of the current script
  - `$HTTP_ACCEPT` : list of MIME types supported by the client
  - `$HTTP_USER_AGENT` : client browser signature
  - `$REMOTE_ADDR` : client IP address
  - `$QUERY_STRING` : string in URL format containing parameters passed to the current page
  - `$HTTP_REFERER` : The URL of the source that returned the client to the current page (by analyzing it, we can know the search engine used as well as the keywords entered by the user, if it actually comes from an engine of research, makes it possible to evaluate the quality of the referencing of a website)

# This lecture covers

- tags
- display text
- comments
- variables
- constants
- control instructions
- stop a script
- arrays
- Associative arrays
- functions
- date() function
- Passing variables with get and post
- Include external files
- Dynamic functions
- Environment variables
- **Environment constants**

# Constants

- script-specific PHP constants
  - can not be redefined
  - useful for handling internal script errors
- `__FILE__`
  - name of the current file
- `__LINE__`
  - current line number
- `PHP_VERSION`
  - PHP version
- `PHP_OS`
  - OS of the web server
- `TRUE`
- `FALSE`

# Constants

- declared predominantly
  - specify to the server's php interpreter what level of rigor apply to faults
- `E_ERROR`
  - error other than "parse error" that can not be corrected
- `E_WARNING`
  - denotes a context in which the PHP finds that something is wrong, alerts retrieved by the script itself
- `E_PARSE`
  - analyzer denotes an invalid syntactic form, impossible correction
- `E_NOTICE`
  - may be error, example: attempt to access a variable that is not yet assigned
- `E_ALL`
  - all constants
- `E_*`
  - used with `error_reporting()`
  - any errors and problems notified



# Constants

Constant	Value
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE

- by default PHP is permissive
  - allows use of variables before creation
  - implicit cast
- `error_reporting($nbr)`
  - set the PHP error report level
  - makes it possible to force a greater severity

# error\_reporting()



The image shows a code editor window with a file named `error_reporting.php` and a web browser window displaying the output of the script.

**Code Editor (error\_reporting.php):**

```
1 <html>
2 <body>
3 <h2>Testing Error Reporting</h2>
4 <?php
5     // Show errors
6     ini_set('display_errors',1);
7
8     // Adjust error reporting
9     error_reporting(E_ALL);
10
11    // Create errors
12    foreach($var as $v){}
13    $result=1/0;
14 ?>
15 </body>
16 </html>
```

**Web Browser (localhost/ch2/error\_reporting.php):**

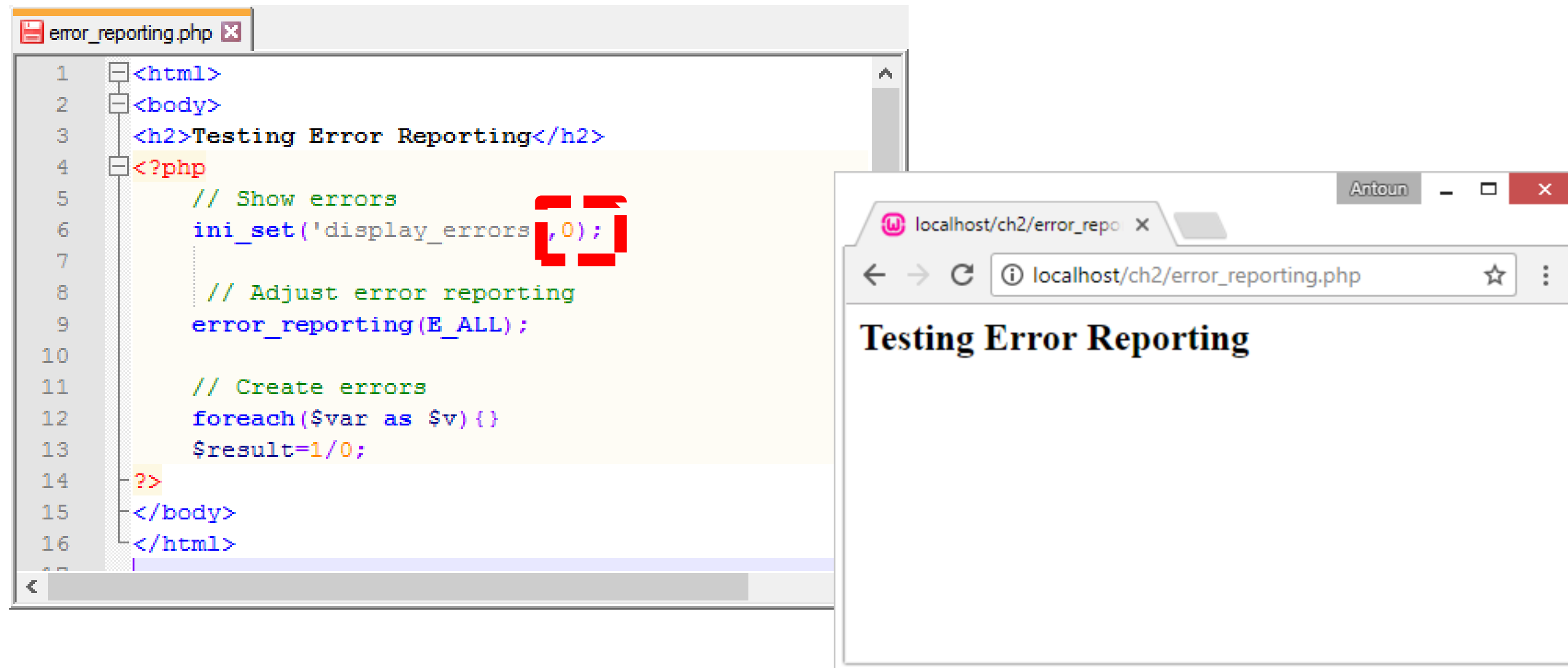
**Testing Error Reporting**

**Notice:** Undefined variable: var in  
D:\wamp\www\ch2\error\_reporting.php on line 12

**Warning:** Invalid argument supplied for foreach() in  
D:\wamp\www\ch2\error\_reporting.php on line 12

**Warning:** Division by zero in  
D:\wamp\www\ch2\error\_reporting.php on line 13

# error\_reporting()



errors are still there but not displayed