

COMP284 Scripting Languages

Lecture 10: PHP (Part 2)

Handouts (8 on 1)

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Scalar types

Integers and Floating-point numbers

Integers and Floating-point numbers: NAN and INF

NAN and INF can be compared with each other and other numbers using **equality** and **comparison operators**:

NAN == NAN ~ FALSE	NAN === NAN ~ FALSE	NAN == 1 ~ FALSE
INF == INF ~ FALSE	INF === INF ~ TRUE	INF == 1 ~ FALSE
NAN < NAN ~ TRUE	INF < INF ~ TRUE	1 < INF ~ TRUE
NAN < INF ~ TRUE	INF < NAN ~ TRUE	INF < 1 ~ FALSE
NAN < 1 ~ TRUE	1 < NAN ~ TRUE	

In PHP 5.3 and earlier versions, `INF == INF` returns **FALSE**

In PHP 5.4 and later versions, `INF == INF` returns **TRUE**

COMP284 Scripting Languages

Lecture 10

Slide L10 – 4

Contents

- 1 Scalar types
 - Integers and Floating-point numbers
 - Exceptions and error handling
 - Booleans
 - Strings
- 2 Compound types
 - Arrays
 - Foreach-loops
 - Array functions
- 3 Printing

COMP284 Scripting Languages

Lecture 10

Slide L10 – 1

Scalar types

Integers and Floating-point numbers

Integers and Floating-point numbers: NAN and INF

- PHP provides three functions to test whether a value is or is not NAN, INF or -INF:
 - `bool is_nan(value)`
returns TRUE iff `value` is NAN
 - `bool is_infinite(value)`
returns TRUE iff `value` is INF or -INF
 - `bool is_finite(value)`
returns TRUE iff `value` is neither NAN nor INF/-INF
- In conversion to a **boolean value**, both NAN and INF are converted to **TRUE**
- In conversion to a **string**, NAN converts to 'NAN' and INF converts to 'INF'

Scalar types

Integers and Floating-point numbers

Integers and Floating-point numbers

- PHP distinguishes between
 - **integer numbers** 0 2012 -10 126.998
 - **floating-point numbers** 1.25 256.0 -12e19 2.4e-10
- PHP supports a wide range of pre-defined mathematical functions
 - `abs(number)` absolute value
 - `ceil(number)` round fractions up
 - `floor(number)` round fractions down
 - `round(number [,prec,mode])` round fractions
 - `log(number [,base])` logarithm
 - `rand(min,max)` generate an integer random number
 - `sqrt(number)` square root
- PHP provides a range of pre-defined number constants including
 - `M_PI` 3.14159265358979323846
 - `NAN` 'not a number'
 - `INF` 'infinity'

COMP284 Scripting Languages

Lecture 10

Slide L10 – 2

Exceptions and error handling

PHP is as follows:

```
try {  
    // try  
} catch (Exception $e) {  
    ... handle the exception here using $e // catch  
}
```

- Errors must be dealt with by an **error handling function** ('Division by zero' produces an error not an exception)
- One possible approach is to let the error handling function turn **errors** into **exceptions**

```
function exception_error_handler($errno, $errstr,  
    $errfile, $errline ) {  
    throw new RuntimeException($errstr, $errno,  
        0, $errfile, $errline);  
}  
set_error_handler("exception_error_handler");
```

<http://www.php.net/manual/en/class.errorexception.php>

COMP284 Scripting Languages

Lecture 10

Slide L10 – 6

Scalar types

Integers and Floating-point numbers

Integers and Floating-point numbers: NAN and INF

The constants NAN and INF are used as **return values** for some applications of mathematical functions that do not return a number

- `log(0)` returns -INF (negative 'infinity')
- `sqrt(-1)` returns NAN ('not a number')

In contrast

- `1/0` returns **FALSE** and produces an error message
 - `0/0` returns **FALSE** and produces an error message
- and execution of the script continues!

In PHP 7

- `1/0` returns **INF** and produces an **error message**
 - `0/0` returns **NAN** and produces an **error message**
- and execution of the script continues!

COMP284 Scripting Languages

Lecture 10

Slide L10 – 3

Scalar types

Booleans

Booleans

- Unlike Perl, PHP does have a **boolean datatype** with constants **TRUE** and **FALSE** (case insensitive)
- PHP offers the same **short-circuit boolean operators** as Java and Perl:
 - `&&` (**conjunction**)
 - `||` (**disjunction**)
 - `!` (**negation**)
- Alternatively, **and** and **or** can be used instead of `&&` and `||`, respectively
- However, **not** is **not** a PHP operator
- The **truth tables** for these operators are the same as for Perl
- Remember that `&&` and `||` are **not** commutative, that is,
(A `&&` B) is not the same as (B `&&` A)
(A `||` B) is not the same as (B `||` A)

COMP284 Scripting Languages

Lecture 10

Slide L10 – 7

<div>Scalar typesBooleans</div> <div>Type conversion to boolean</div> <p>When converting to boolean, the following values are considered FALSE:</p> <ul style="list-style-type: none"> the boolean FALSE itself the integer 0 (zero) the float 0.0 (zero) the empty string, and the string '0' an array with zero elements an object with zero member variables (PHP 4 only) the special type NULL (including unset variables) SimpleXML objects created from empty tags <p>Every other value is considered TRUE (including any resource)</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 8</div>	<div>Compound typesArrays</div> <div>Arrays</div> <ul style="list-style-type: none"> It is possible to omit the keys when using the array construct: <pre>\$arr3 = array("Peter", "Paul", "Mary");</pre> <p>The values given in array will then be associated with the natural numbers 0, 1, ...</p> All the keys of an array can be retrieved using array_keys(\$array1) <ul style="list-style-type: none"> returns a natural number-indexed array containing the keys of \$array1 All the values of an array can be retrieved using array_values(\$array1) <ul style="list-style-type: none"> returns a natural number-indexed array containing the values stored in \$array1 <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 12</div>
<div>Scalar typesStrings</div> <div>Strings</div> <ul style="list-style-type: none"> PHP supports both single-quoted and double-quoted strings PHP also supports heredocs as a means to specify multi-line strings <p>The only difference to Perl is the use of <code><<<</code> instead of <code><<</code> in their definition:</p> <pre><<<identifier here document identifier</pre> <ul style="list-style-type: none"> identifier might optionally be surrounded by double-quotes identifier might also be surrounded by single-quotes, making the string a nowdoc in PHP terminology <pre>print '<html> <head><title>Multi-Line String</title></head> </head><body>Some text</body> </html> EOF; EOF;</pre> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 9</div>	<div>Compound typesArrays</div> <div>Arrays</div> <ul style="list-style-type: none"> An individual array element can be accessed via its key Accessing an undefined key produces an error message and returns NULL <pre>\$arr1 = array(1 => "Peter", 3 => 2009, "a" => 101); print "'a':".\$arr1["a"]."\n"; 'a': 101 print "'b':".\$arr1["b"]."\n"; PHP Notice: Undefined index: b in <file> on line <lineno> 'b': // \$arr1["b"] returns NULL \$arr1['b'] = 102; print "'b':".\$arr1["b"]."\n"; 'b': 102</pre> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 10</div>
<div>Scalar typesStrings</div> <div>Strings</div> <ul style="list-style-type: none"> Variable interpolation is applied to double-quoted strings (with slight differences to Perl) The string concatenation operator is denoted by <code>.</code> (as in Perl) Instead of Perl's string multiplication operator <code>'x'</code> there is string <code>str_repeat(string_arg, number)</code> There are no built-in HTML shortcuts in PHP <pre>\$title = "String_Multiplication"; \$string = "<p>I shall not repeat myself.<p>\n"; print "<!DOCTYPE html>\n<html><head><title>\$title</title></head><body>".str_repeat(\$string, 3)."</body></html>"; <!DOCTYPE html> <html><head><title>String Multiplication</title></head><body><p>I shall not repeat myself.<p> <p>I shall not repeat myself.<p> <p>I shall not repeat myself.<p> </body></html></pre> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 10</div>	<div>Compound typesArrays</div> <div>Arrays</div> <ul style="list-style-type: none"> PHP provides a foreach-loop construct to 'loop' through the elements of an array Syntax and semantics is slightly different from that of the corresponding construct in Perl <pre>foreach (array as \$value) statement foreach (array as \$key => \$value) statement</pre> <ul style="list-style-type: none"> array is an array expression \$key and \$value are two variables, storing a different key-value pair in array at each iteration of the foreach-loop We call \$value the foreach-variable foreach iterates through an array in the order in which elements were defined <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 15</div>
<div>Compound typesArrays</div> <div>Arrays</div> <ul style="list-style-type: none"> PHP only supports associative arrays (hashes), simply called arrays PHP arrays are created using the array construct or, since PHP 5.4, <code>[...]</code>: <pre>array(key => value, ...) [key => value, ...]</pre> <p>where key is an integer or string and value can be of any type, including arrays</p> <pre>\$arr1 = [1 => "Peter", 3 => 2009, "a" => 101]; \$arr2 = array(200846369 => array("name" => "Jan_Olsen", "COMP101" => 69, "COMP102" => 52));</pre> The size of an array can be determined using the count function: <pre>int count(array [, mode]) print count(\$arr1); // prints 3 print count(\$arr2); // prints 1 print count(\$arr2,1); // prints 4</pre> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 11</div>	<div>Compound typesForeach-loops</div> <div>Arrays: foreach-loop</div> <ul style="list-style-type: none"> PHP provides a foreach-loop construct to 'loop' through the elements of an array Syntax and semantics is slightly different from that of the corresponding construct in Perl <pre>foreach (array as \$value) statement foreach (array as \$key => \$value) statement</pre> <ul style="list-style-type: none"> array is an array expression \$key and \$value are two variables, storing a different key-value pair in array at each iteration of the foreach-loop We call \$value the foreach-variable foreach iterates through an array in the order in which elements were defined <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 15</div>

<div>Compound typesForeach-loops</div> <div>Arrays: foreach-loop</div> <p><code>foreach</code> iterates through an array in the order in which elements were defined</p> <p>Example 1:</p> <pre>foreach (array("Peter", "Paul", "Mary") as \$key => \$value) print "The array maps \$key to \$value\n";</pre> <p>The array maps 0 to Peter The array maps 1 to Paul The array maps 2 to Mary</p> <p>Example 2:</p> <pre>\$arr5[2] = "Marry"; \$arr5[0] = "Peter"; \$arr5[1] = "Paul"; // 0 => 'Peter', 1 => 'Paul', 2 => 'Marry' foreach (\$arr5 as \$key => \$value) print "The array maps \$key to \$value\n";</pre> <p>The array maps 2 to Mary The array maps 0 to Peter The array maps 1 to Paul</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 16</div>	<div>Compound typesArray functions</div> <div>Array functions</div> <p>PHP has no <code>stack</code> or <code>queue</code> data structures, but has <code>stack</code> and <code>queue</code> functions for <code>arrays</code>:</p> <ul style="list-style-type: none"> <code>array_push(\$array, \$value1, \$value2, ...)</code> appends one or more elements at the end of the end of an array variable; returns the number of elements in the resulting array <code>array_pop(\$array)</code> extracts the last element from an array and returns it <code>array_shift(\$array)</code> shift extracts the first element of an array and returns it <code>array_unshift(\$array, \$value1, \$value2, ...)</code> inserts one or more elements at the start of an array variable; returns the number of elements in the resulting array <p>Note: <code>\$array</code> needs to be a <code>variable</code></p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 20</div>
<div>Compound typesForeach-loops</div> <div>Arrays: foreach-loop</div> <p>Does changing the value of the <code>foreach-variable</code> change the element of the list that it currently stores?</p> <p>Example 3:</p> <pre>\$arr6 = array("name" => "Peter", "year" => 2009); foreach (\$arr6 as \$key => \$value) { print "The array maps \$key to \$value\n"; \$value .= " - modified"; // Changing \$value } print "\n"; foreach (\$arr6 as \$key => \$value) print "The array maps \$key to \$value\n";</pre> <p>The array maps name to Peter The array maps year to 2009</p> <p>The array now maps name to Peter The array now maps year to 2009</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 17</div>	<div>Printing</div> <div>Printing</div> <p>In PHP, the default command for generating output is <code>echo</code></p> <ul style="list-style-type: none"> <code>void echo(\$arg1)</code> <code>void echo \$arg1, \$arg2, ...</code> Outputs all arguments No parentheses are allowed if there is more than one argument More efficient than <code>print</code> (and therefore preferred) <p>Additionally, PHP also provides the functions <code>print</code>, and <code>printf</code>:</p> <ul style="list-style-type: none"> <code>int print(\$arg)</code> Outputs its argument Only one argument is allowed! Returns value 1 <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 18</div>
<div>Compound typesForeach-loops</div> <div>Arrays: foreach-loop</div> <ul style="list-style-type: none"> In order to modify array elements within a <code>foreach-loop</code> we need use a <code>reference</code> <pre>foreach (array as &\$amp;\$value) statement unset(\$value); foreach (array as \$key => &\$amp;\$value) statement unset(\$value);</pre> <ul style="list-style-type: none"> In the code schemata above, <code>&\$value</code> is a variable whose value is stored at the same location as an array element Note that PHP does not allow the <code>key</code> to be a reference The <code>unset</code> statement is important to return <code>\$value</code> to being a 'normal' variable <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 18</div>	<div>Printing</div> <div>Printing</div> <ul style="list-style-type: none"> <code>int printf(\$format, \$arg1, \$arg2, ...)</code> Produces output according to <code>\$format</code> Parentheses are necessary Returns the length of the outputted string <p>See http://www.php.net/manual/en/function.sprintf.php for details</p> <ul style="list-style-type: none"> <code>int printf(\$format, \$arg1, \$arg2, ...)</code> Produces output according to <code>\$format</code> Parentheses are necessary Returns the length of the outputted string <p>Important: In contrast to Perl, a PHP array cannot take the place of a list of arguments</p> <pre>printf("%2d apples %2d oranges\n", array(5,7));</pre> <p>produces an error message</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 22</div>
<div>Compound typesForeach-loops</div> <div>Arrays: foreach-loop</div> <p>In order to modify array elements within a <code>foreach-loop</code> we need use a <code>reference</code></p> <p>Example:</p> <pre>\$arr6 = array("name" => "Peter", "year" => 2009); foreach (\$arr6 as \$key => &\$amp;\$value) { // Note: reference! print "The array maps \$key to \$value\n"; \$value .= " - modified"; } unset(\$value); // Remove the reference from \$value print "\n"; foreach (\$arr6 as \$key => \$value) print "The array maps \$key to \$value\n";</pre> <p>The array maps name to Peter The array maps year to 2009</p> <p>The array now maps name to Peter - modified The array now maps year to 2009 - modified</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 19</div>	<div>Printing</div> <div>Printing</div> <ul style="list-style-type: none"> <code>string vsprintf(\$format, array)</code> Returns a string produced according to the formatting string <code>\$format</code> Identical to <code>sprintf</code> but accepts an <code>array</code> as argument Parentheses are necessary <ul style="list-style-type: none"> <code>int vprintf(\$format, array)</code> Produces output according to <code>\$format</code> Identical to <code>printf</code> but accepts an <code>array</code> as argument Parentheses are necessary <pre>vprintf("%2d apples %2d oranges\n", array(5,7));</pre> <p>5 apples 7 oranges</p> <div>COMP284 Scripting LanguagesLecture 10Slide L10 – 23</div>

Revision

Read

- Chapter 6: PHP Arrays

of

R. Nixon:

[Learning PHP, MySQL, and JavaScript.](#)

O'Reilly, 2009.

- <http://uk.php.net/manual/en/language.types.boolean.php>
- <http://uk.php.net/manual/en/language.types.integer.php>
- <http://uk.php.net/manual/en/language.types.float.php>
- <http://uk.php.net/manual/en/language.types.string.php>
- <http://uk.php.net/manual/en/language.types.array.php>
- <http://uk.php.net/manual/en/control-structures.foreach.php>

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro