INFO I-CE9224: Introduction to PHP Programming

Session 4
June 27, 2012

Resources

http://davehauenstein.com/nyu/INFO1-CE9224-2012-Summer

Username: nyuscps
Password: $nyuscps$

Class 4 Agenda

- Homework 1 and Lab 2 Review
- PHP Language Basics Part 4
  - Loops (while, do...while, for, foreach)
  - Multidimensional arrays (nested arrays)
  - Array Functions
  - Intro to User Defined Functions
- Lab Assignment 3

HW 1 and Lab 2

Review

Loops

Looping

Problem:

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich'
);

echo $andriodVersions[0];
echo $andriodVersions[1];
echo $andriodVersions[2];
```

Looping

Problem:

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich',
    'Jelly Bean',
);

echo $andriodVersions[0];
echo $andriodVersions[1];
echo $andriodVersions[2];
echo $andriodVersions[3];
```

1

2

Looping

Solution:

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich',
    'Jelly Bean',
);

for ($x=0; $x < count($andriodVersions); $x++) {
    echo $andriodVersions[$x] . '<br />';
}
```

Looping

- Run a block of code over and over again.
- Stop running block of code once condition is met.
- Loops evaluate a condition similar to the way that if and switch statements evaluate one.
- If some condition is true, the loop will continue to run.
- Once that condition becomes false, the loop will stop.
- An iteration can be skipped (continue).
- A loop can be broken out of (break).

Looping:
Real World Example

1. A form asks a user to list their 10 favorite musicians.
2. For each musician, we must check if they exist in our music database.
3. If it doesn't exist in our database, add it.
4. If it does exist, skip this iteration, move on to the next musician.

Loops

- *while* loops
- *do... while* loops
- *for* loops
- *foreach* loops

while Loop

```
while ( expression ) {
    // code here...
}
```

while Loop

```php
<?php

$products = 10;
while ($products > 0) {
    echo "Selling a product...";
    $products--;
    echo "There are {$products} left.";
}

echo 'All of our products are gone';
```

while Loop

Example: Reading a file line-by-line

```php
<?php

$handle = fopen("/path/to/file.txt", "r");
while (($line = fgets($handle, 4096)) !== false) {
    echo $buffer;
}
```

do... while Loop

```
do {
    // code will get executed at least once.
} while ( expression );
```

do... while Loop

```php
<?php

do {
    if (!isset($products)) {
        $products = 10;
    }
    echo "Sell<br />";
    $products--;
    echo "{$products} left<br /><br />";
} while ($products > 0);
```

for Loop

```
for ( expression1, expression2, expression3 ) {
    // code here...
}
```

Typically used when you
know how many times
you need to loop.

```php
<?php

for ($x = 0; $x < 10; $x++) {
    echo 'Iteration: ' . $x . '<br />';
}

/*
Prints out:

Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
Iteration: 6
Iteration: 7
Iteration: 8
Iteration: 9
*/
```

for Loop

```
for ( expression1, expression2, expression3 ) {
    // code here...
}
```

- expression1: *initializer* - Run only once. Typically a counter.
- expression2: *loop test* - If this is true, the loop continues, if it's false, it breaks out.
- expression3: *counting* - Ran after each iteration of the loop, usually used to change the counter.

for Loop

```php
<?php

for ($x = 0; $x < 10; $x++) {
    echo 'Iteration: ' . $x . '<br />';
}

/*
Prints out:

Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
Iteration: 6
Iteration: 7
Iteration: 8
Iteration: 9
*/
```

for Loop

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich',
    'Jelly Bean',
);

$num = count($andriodVersions);

for ($x=0; $x < $num; $x++) {
    echo $andriodVersions[$x] . '<br />';
}
```

for Loop

Notice:
count() is done before loop

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich',
    'Jelly Bean',
);

$num = count($andriodVersions);

for ($x=0; $x < $num; $x++) {
    echo $andriodVersions[$x] . '<br />';
}
```

for vs. *while* Loop

```php
<?php

$andriodVersions = array(
    'Froyo',
    'Gingerbread',
    'Ice Cream Sandwich',
    'Jelly Bean',
);

$counter = 0;
$num     = count($andriodVersions);

while ($counter < $num) {
    echo $andriodVersions[$counter] . '<br />';
    $counter++;
}
```

Guessing a random number: *break*

```php
<?php

$min  = 0;
$max  = 100;
$rand = rand($min, $max);

for ($x = $min; $x < $max; $x++) {
    if ($x == $rand) {
        echo "Found random: $x";
        break;
    }
}
```

Finding dividends: *continue*

```php
<?php

$num = 3680;

for ($x = 1; $x < $num; $x++) {
    if ($num % $x !== 0) {
        continue;
    }

    echo "$num is evenly divisible by $x<br />";
}
```

foreach loops and looping over Arrays

Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);
```

Lists

```php
$phones = array(
    'iPhone'        => 'Apple',
    'Galaxy Nexus' => 'Google',
    'Lumia 900'     => 'Nokia',
);
```

Dictionaries

Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);

// IS THE SAME AS:

$phones = array(
    0 => 'iPhone',
    1 => 'Galaxy Nexus',
    2 => 'Lumia 900',
);
```

foreach Loop Back to Arrays

- Special kind of loop used only with arrays (and special objects).
- Retrieve just the value of the element, or both the key (or index) and the value.
- Will automatically break out of itself once it's gone through every element in the array.
- Probably one of the most used constructs in the language.

foreach Loop Back to Arrays

```
foreach ( $array as $value) {
    echo $value;
}
```

Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);
```

Lists

```php
$phones = array(
    'iPhone'        => 'Apple',
    'Galaxy Nexus' => 'Google',
    'Lumia 900'     => 'Nokia',
);
```

Dictionaries

Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);

// IS THE SAME AS:

$phones = array(
    0 => 'iPhone',
    1 => 'Galaxy Nexus',
    2 => 'Lumia 900',
);
```

foreach Loop Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);

foreach ($phones as $phone) {
    echo $phone . "<br />";
}

// Prints out:
//    iPhone
//    Galaxy Nexus
//    Lumia 900
```

foreach Loop Back to Arrays

```php
$phones = array(
    'iPhone'         => 'Apple',
    'Galaxy Nexus' => 'Google',
    'Lumia 900'     => 'Nokia',
);

foreach ($phones as $phone) {
    echo $phone . "<br />";
}

// Prints out:
//    Apple
//    Google
//    Nokia
```

foreach Loop
Back to Arrays

```
foreach ( $array as $key => $value) {
    echo $value;
}
```

foreach Loop
Back to Arrays

```php
$phones = array(
    'iPhone',
    'Galaxy Nexus',
    'Lumia 900',
);

foreach ($phones as $index => $phone) {
    echo $index . ': ' . $phone . "<br />";
}

// Prints out:
//    0: iPhone
//    1: Galaxy Nexus
//    2: Lumia 900
```

foreach Loop Back to Arrays

```php
$phones = array(
    'iPhone'         => 'Apple',
    'Galaxy Nexus' => 'Google',
    'Lumia 900'     => 'Nokia',
);

foreach ($phones as $phone => $brand) {
    echo $phone . " by: " . $brand . "<br />";
}

// Prints out:
//    iPhone by: Apple
//    Galaxy Nexus by: Google
//    Lumia 900 by: Nokia
```

Example: *foreach* Loop

Displaying all of the values in
the $_SERVER super global.

```php
foreach($_SERVER as $key => $value) {
    echo "{$key}: {$value}<br />";
}
```

```php
<?php
    if($_POST) {
        foreach($_POST as $field => $value) {
            if ($field == 'Submit') continue;

            echo str_replace('_', ' ', $field) . ': ';
            echo $value . '<br />';
        }
    }
?>
<form action="self" method="post">
    <ul>
        <li>
            <label for="product_name">Product Name</label>
            <input type="text" name="product_name" id="product_name" />
        </li>
        <li>
            <label for="price">Price</label>
            <input type="text" name="price" id="price" />
        </li>
        <li>
            <input type="submit" name="Submit" value="Submit" id="submit" />
        </li>
    </ul>
</form>
```

Some Array Functions

Array Functions

- count()
- sort() and rsort()
- asort() and arsort()
- ksort() and krsort()
- implode() and explode()

Full List: http://php.net/manual/en/ref.array.php

count()

Returns the number of elements in an array.

```
$cars = array(‘acura’, ‘ford’, ‘vw’);
echo count($cars); // prints (int) 3
```

sort() and rsort()

```php
<?php

$numbers = array(
    145,
    92,
    194,
);

sort($numbers);
print_r($numbers);

// Array
// (
//     [0] => 92
//     [1] => 145
//     [2] => 194
// )
```

```php
<?php

$letters = array(
    'c',
    'r',
    'a',
);

rsort($letters);
print_r($letters);

// Array
// (
//     [0] => r
//     [1] => c
//     [2] => a
// )
```

asort() and arsort()

```php
<?php

$phones = array(
    'Galaxy Nexus' => 'Google',
    'Lumia 900'    => 'Nokia',
    'iPhone'       => 'Apple',
);

asort($phones);
print_r($phones);

// Array
// (
//   [iPhone] => Apple
//   [Galaxy Nexus] => Google
//   [Lumia 900] => Nokia
// )
```

ksort() and krsort()

```php
<?php

$phones = array(
    'Galaxy Nexus' => 'Google',
    'Lumia 900'    => 'Nokia',
    'iPhone'       => 'Apple',
);

ksort($phones);
print_r($phones);

// Array
// (
//    [Galaxy Nexus] => Google
//    [Lumia 900] => Nokia
//    [iPhone] => Apple
// )
```

implode()

Convert an array into a string using some string delimiter.

```php
<?php

$fields = array(
    'name'       => 'Dave',
    'profession' => 'Developer',
    'city'       => 'New York',
);

echo implode("<br />\n", $fields);

// Prints out:
//
// Dave<br />
// Developer<br />
// New York
```

explode()

Convert a string into an array using some string delimiter.

```php
<?php

$sentence = 'It is quite a nice day';
print_r(explode(" ", $sentence));

// Array
// (
//     [0] => It
//     [1] => is
//     [2] => quite
//     [3] => a
//     [4] => nice
//     [5] => day
// )
```

Multidimensional Arrays

Multidimensional Arrays

- An array can contain values of any type; object, string, int, etc...
- When an array has values that are other arrays, it's called multidimensional (or nested arrays)
- If a containing array also contains an array, the top level array is called a three-dimensional array.
- This is a very powerful aspect of PHP.

Multidimensional Arrays

```php
<?php

$books = array(
    array(
        'title'    => 'The Grapes of Wrath',
        'author'   => 'John Steinbeck',
        'pubYear' => 1939,
    ),
    array(
        'title'    => 'The Trial',
        'author'   => 'Franz Kafka',
        'pubYear' => 1925,
    ),
);

print_r($books);
```

Multidimensional Arrays

```
Array
(
    [0] => Array
        (
            [title] => The Grapes of Wrath
            [author] => John Steinbeck
            [pubYear] => 1939
        )

    [1] => Array
        (
            [title] => The Trial
            [author] => Franz Kafka
            [pubYear] => 1925
        )

)
```

Multidimensional Arrays

Accessing Elements in Multidimensional arrays

```
echo $books[0]['title'];
```

prints out: The Grapes of Wrath

Multidimensional Arrays

```php
<?php

$bestSellers = array(
    'John Steinbeck' => array(
        'title'   => 'The Grapes of Wrath',
        'pubYear' => 1939,
    ),
    'Franz Kafka' => array(
        'title'   => 'The Trial',
        'pubYear' => 1925,
    ),
);

print_r($bestSellers);
```

Multidimensional Arrays

```
Array
(
    [John Steinbeck] => Array
        (
            [title] => The Grapes of Wrath
            [pubYear] => 1939
        )

    [Franz Kafka] => Array
        (
            [title] => The Trial
            [pubYear] => 1925
        )

)
```

Multidimensional Arrays

Accessing Elements in Multidimensional arrays

```
echo $bestSellers['Franz Kafka']['title'];
```

prints out: The Trial

Nested Loops and Looping Through Multidimensional Arrays

```php
$bestSellers = array(
    'John Steinbeck' => array(
        'title'   => 'The Grapes of Wrath',
        'pubYear' => 1939,
    ),
    'Franz Kafka' => array(
        'title'   => 'The Trial',
        'pubYear' => 1925,
    ),
);

foreach ($bestSellers as $author => $details) {
    echo "<strong>{$author}:</strong><br />";

    foreach ($details as $key => $value) {
        echo "<strong>{$key}:</strong>{$value}<br />";
    }

    echo "<br />";
}
```

John Steinbeck:
title: The Grapes of Wrath
pubYear:1939

Franz Kafka:
title: The Trial
pubYear:1925

```php
$bestSellers = array(
    'John Steinbeck' => array(
        'title'    => 'The Grapes of Wrath',
        'pubYear'  => 1939,
        'language' => 'English',
    ),
    'Franz Kafka' => array(
        'title'    => 'The Trial',
        'pubYear'  => 1925,
        'language' => 'German',
    ),
);

foreach ($bestSellers as $author => $details) {
    if ($details['language'] !== 'English') {
        continue;
    }

    echo "<strong>{$author}:</strong><br />";

    foreach ($details as $key => $value) {
        echo "<strong>{$key}:</strong>{$value}<br />";
    }

    echo "<br />";
}
```

User Defined Functions

User Defined Functions

- Self contained block of code to perform a specific task.
- Modular: Once defined, can be called many times, anywhere in a script.
- Avoid duplication of code.
- Easier to eliminate errors.
- Help break down a big project into smaller pieces.

User Defined Functions

- Defining your own functions
- Return values
- Function arguments
- Optional arguments

Defining your own functions

```php
// Defining your own function
function getFullName ($first, $last) {
    $first = ucfirst($first);
    $last  = ucfirst($first);

    return $first . ' ' . $last;
}

// calling your defined function
echo getFullName('dave', 'hauenstein');

// prints out: Dave Hauenstein
```

Defining your own functions

```php
// Defining your own function
function add ($num1, $num2) {
    if (!is_int($num1) || !is_int($num2)) {
        return 'Could not add!';
    }

    return $num1 + $num2;
}

// calling your defined function
echo add(4, 5); // prints out: 9
echo add('4', 5); // prints out: Could not add!
```

Defining your own functions

```php
// Defining your own function
function getList (array $data) {
    $list = '<ul>';
    foreach($data as $value) {
        $list .= "<li>$value</li>";
    }
    $list .= "</ul>";

    return $list;
}

$list = array(
    'Ford',
    'Chevy',
    'Dodge',
);
// calling your defined function
echo getList($list);
```

Defining your own functions

```php
// Defining your own function
function getList (array $data, $id = '') {
    if ($id) {
        $list = '<ul id="' . $id . '">';
    } else {
        $list = '<ul>';
    }

    foreach($data as $value) {
        $list .= "<li>$value</li>";
    }
    $list .= "</ul>";

    return $list;
}

$list = array(
    'Ford',
    'Chevy',
    'Dodge',
);
// calling your defined function
echo getList($list, 'us-car-companies');
```

Lab Assignment!