Lecture 12

Mapping, Map Reduction
Summing integers,
Map Reduce & Strings
PHP Objects

* Course logo spider web photograph from Morguefile openstock photograph by Gabor Karpati, Hungary.
Example 02

- Reduction trumps iteration!
- Two arguments,
  - First builds result,
  - Second is current array element.

```php
$ar = array(2, 4, 6, 8);
function runsum($s, $next) {
    $res = $s + $next;
    echo "Running sum: $s ... next value: $next ... returns: $res<br/>
";
    return $res;
}
echo "<p>The array is: </p>
";
print_r($ar);
echo "<p>About to carry out the array reduction.<br/>
";
$foo = array_reduce($ar, "runsum", 0);
echo "<p>The results is: $foo </p>
";
```
Example 02

Array reduce, first add integers.

The array is:

```
```

About to carry out the array reduction.

Running sum: 0 ... next value: 2 ... returns: 2
Running sum: 2 ... next value: 4 ... returns: 6
Running sum: 6 ... next value: 6 ... returns: 12
Running sum: 12 ... next value: 8 ... returns: 20

The results is: 20
Here is how functions are defined.

```php
function runsum($s, $next) {
    $res = $s + $next;
    echo "Running sum: $s ... next value: $next ... 
    return $res;
}
```

Feature you have come to expect:

- A name: “runsum”
- Arguments – no typing
- Code body
- Explicit return value – no typing again.
The Naming of Functions

- Consider Carefully this Code

```
$foo = array_reduce($ar, 'runsum', 0);
```

- What is the second argument?
  - Is it a string literal?
  - Is it a function?

When you are comfortable saying yes to be both and understand why that answer makes sense you will be on your way to understanding `array_reduce` in particular and the idea of a run-time symbol table more generally.
Mapping In General

- In C, C++, Java and the like there is a STRONG bias in favor of loops.
- There has always been an alternative view in CS ...

> In LISP, functional arguments are extremely useful. A very important function with a functional argument is maplist. Its M-expression definition is


- PHP supports Mapping!!
Map runsum as a Figure

\[
\begin{align*}
\text{[0]} & \Rightarrow 2 \\
\text{[1]} & \Rightarrow 4 \\
\text{[2]} & \Rightarrow 6 \\
\text{[3]} & \Rightarrow 8 \\
\text{Accumulation variable } & s \\
\text{Array element } & \text{next}
\end{align*}
\]
$foo = array_reduce($ar, 'runsum', 42);

The array is:
About to carry out the array reduction.
Running sum: 42 ... next value: 2 ... returns: 44
Running sum: 44 ... next value: 4 ... returns: 48
Running sum: 48 ... next value: 6 ... returns: 54
Running sum: 54 ... next value: 8 ... returns: 62
The results is: 62
Starting with 42

42 + 2

44 + 4

48 + 6

54 + 8

[0] => 2

[1] => 4

[2] => 6

[3] => 8

62
array_reduce is a function.
It does return a result.

array_reduce

(PHP 4 >= 4.0.5, PHP 5, PHP 7)
array_reduce — Iteratively reduce the array to a single value using a callback function

Description

mixed array_reduce ( array $array, callable $callback [, mixed $initial = NULL ] )

array_reduce() applies iteratively the callback function to the elements of the array, so as to reduce the array to a single value.
Example 03

- The combination function may be created in place, not named.
- The short if-then-else format is handy for the return.
- Second example is better.

```php
$ar1 = array('cat', 'dog', 'fish', 'hampster', 'cats', 'turtles');
echo '<p>Here is the array: </p>' . 
print_r($ar1); 
echo '</p>

$foo = array_reduce($ar1, create_function('$a,$b', 'return "$a and $b";'));
$bar = array_reduce($ar1, create_function('$a,$b', 'return is_null($a) ? "$b" : "$a and $b";'));

<p>First example has a leading 'and' due to empty string initialization.</p>
<p>Animals: <?php echo $foo ?></p>
<p>Second example tests for first instance and handled as a special case.</p>
<p>Animals: <?php echo $bar ?></p>
```
Array reduce is useful to build strings. For example joining nouns.

Here is the array:


First example has a leading 'and' due to empty string initialization.

Animals: and cat and dog and fish and hampster and cats and turtles

Second example tests for first instance and handled as a special case.

Animals: cat and dog and fish and hampster and cats and turtles
Why The Leading ‘and’

- In the first case, foo, the initialization value is null the first time the function is called.
- What is returned is null – the empty string for strings – concatenated with ‘and’
- The second example uses the short syntax if-the-else to avoid the problem.
A Function with No Name

Notice the in-place definition of the function passed to `map_reduce`.

```php
array_reduce($ar1, create_function('{$a,$b}', 'return "$a and $b";'));
```

**create_function**

(PHP 4 >= 4.0.1, PHP 5, PHP 7)
create_function — Create an anonymous (lambda-style) function

**Description**

```php
string create_function ( string $args , string $code )
```

Creates an anonymous function from the parameters passed, and returns a unique name for it.
Anonymous functions

Anonymous functions, also known as closures, allow the creation of functions which have no specified name. They are most useful as the value of `callback` parameters, but they have many other uses.

Example #1 Anonymous function example

```php
<?php
    echo preg_replace_callback('~-(\[a-z])~', function ($match) {
        return strtoupper($match[1]);
    }, 'hello-world');
    // outputs helloWorld
?>
```
Here is a portion of Example 3 using the new more general way of creating anonymous functions:

```php
<?php
$ar1 = array('cat', 'dog', 'fish', 'hampster', 'cats', 'turtles');
echo '<p>Here is the array: </p>'."\n".'<p>'."\n";
print_r($ar1);
echo "\n</p>\n";
$foo = array_reduce($ar1, function($a,$b) { return "$a and $b"; });
```

This new format better implements what is commonly called a **closure**. Put simply, closures have everything to do with variable scoping, and the PHP documentation has some nice examples.
Example 04

- PHP is an object oriented language.
- Here is a class for ‘States’.

```php
class state {
    public $name;
    public $capital;

    function __construct($n, $c) {
        $this->name = $n;
        $this->capital = $c;
    }

    function __toString() {
        return "State of $this->name with Capital $this->capital";
    }
}
```
Example 04

```php
$a = array();
$a[] = new state ( 'Colorado', 'Denver' );
$a[] = new state ( 'California', 'Sacramento' );
$a[] = new state ( 'Wyoming', 'Cheyenne' );

foreach ( $a as $s ) {
    echo "<p>$s</p>\n";
    // echo "<p>" . $s->shortName () . "</p>\n";
}
```

There are objects in PHP and arrays of objects are useful!

State of Colorado with Capital Denver

State of California with Capital Sacramento

State of Wyoming with Capital Cheyenne
Object, print thy self

- In Example 4, pay attention to the use of the `__toString()` method.
- Most modern Object Oriented Languages expect a programmer to take responsibility for how an object responds to a print – echo – statement.
- Since so much of PHP is about generating text - HTML – output this is a useful capability.
About Arrays of Objects

```javascript
Array
{
    [0] => state Object
        {
            [name] => Colorado
            [capital] => Denver
        }
    [1] => state Object
        {
            [name] => California
            [capital] => Sacramento
        }
    [2] => state Object
        {
            [name] => Wyoming
            [capital] => Cheyenne
        }
}
```

Recognize This Format?