Lecture 13

Tables from Arrays of Objects
Exploding Strings

Course logo spider web photograph from Morguefile openstock photograph by Gabor Karpati, Hungary.
Sets of objects may be displayed in a table.


<table>
<thead>
<tr>
<th>name</th>
<th>capital</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
</tbody>
</table>

Example 5 combines: Objects, Arrays, HTML Tables
Example 05

- Use objects to manage information
- A state is a singular thing with 3 related data items:

```php
27 class state {
28     public $name;
29     public $capital;
30     public $population;
31     function __construct($n, $c, $p) {
32         $this->name = $n;
33         $this->capital = $c;
34         $this->population = $p;
35     }
36 }
```
Sets of objects may be displayed in a table.


<table>
<thead>
<tr>
<th>name</th>
<th>capital</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
</tbody>
</table>

Punchline: Transform an array of objects to a table
An Array of States

- Create three states in an array
- Note line 41: array holds all the member data for the first state.

```php
37  $a = array ();
38  $a [0] = new state ( 'Colorado', 'Denver', 5029196 );
39  $a [1] = new state ( 'California', 'Sacramento', 37253956 );
40  $a [2] = new state ( 'Wyoming', 'Cheyenne', 563626 );
41  print_r ( get_object_vars ( $a [0] ));
42  echo '<table>' . "\n";
43  echo '<tr>' . "\n";
44  $keys = array_keys ( get_object_vars ( $a [1] ));
45  foreach ( $keys as $k ) {
46      echo '<th>' . $k . ' </th>' . "\n";
47  }
```
More about `get_object_vars`

- With one function you can fully equate object member data with a keyword style array.

```
Array
(
    [name] => Colorado
    [capital] => Denver
    [population] => 5029196
)
```

- Access to the keys is very handy.
Keys Are Column Headers

Here again is a part of the code.
Line 44: Object member data to an array
Line 44: Retrieve keywords from array
Line 45: Iterate through keys
Line 46: One table column header per key.

```php
37 $a [0] = new state ( 'Colorado', 'Denver', 8029190 );
38 $a [1] = new state ( 'California', 'Sacramento', 37253956 );
39 $a [2] = new state ( 'Wyoming', 'Cheyenne', 563626 );
40 print_r ( get_object_vars ( $a [0] ) );
41 echo '<table>' . "\n";
42 echo '  <tr>' . "\n";
43 foreach ( $keys as $k ) {
44   echo '    <th>' . $k . '    </th>' . "\n";
45 }
```
Filling in the Table

Enumerate the values in the table

```php
foreach ( $a as $s ) {
    $vals = array_values ( get_object_vars ( $s ) );
    echo '   <tr>' . "\n";
    foreach ( $vals as $v ) {
        echo '       <td>' . $v . ' </td>' . "\n";
    }
    echo '  </tr>' . "\n";
}
```

Upper and lower versions function the same. Lower avoids intermediate variable.
<style>
  table {
    border: medium solid #7f7f7f;
    margin: 8px 16px 8px 16px;
  }
  th, td {
    text-align: right;
    padding: 0px 6px 0px 6px;
  }
  th {
    border-bottom: thin solid #7f7f7f
  }
</style>

Array ([name] => Colorado [capital] => Denver [population] => 5029196 )

<table>
<thead>
<tr>
<th>name</th>
<th>capital</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
<tr>
<td>Colorado</td>
<td>Denver</td>
<td>5029196</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>37253956</td>
</tr>
<tr>
<td>Wyoming</td>
<td>Cheyenne</td>
<td>563626</td>
</tr>
</tbody>
</table>
Now Without Loops

- Same goal, build a table from an array of objects.
- But, no loops, just array reduction
  - Nested array reduction at that.
- Many ways this could be done.
- In this example, table header and table data rows first built as strings.
- Then sent to the client, i.e. a succinct echo command inside table tags.
Example 6 Code

```php
37 $a = array ();
38 $a [] = new state ( 'Colorado', 'Denver', 5029196 );
39 $a [] = new state ( 'California', 'Sacramento', 37253956 );
40 $a [] = new state ( 'Wyoming', 'Cheyenne', 563626 );
41 $ths = "<tr>" . array_reduce ( array_keys ( get_object_vars ( $a [1] ) ),
42     function ($s, $n) {
43         return $s . "<th>" . ucfirst($n) . "</th>";
44     } ). "</tr>\n";
45 $tds = array_reduce ( $a, function ($s1, $n1) {
46     return $s1 . "<tr>" .
47         array_reduce ( get_object_vars ( $n1 ),
48             function ($s2, $n2) {
49                 return $s2 . "<td> $n2 </td>";
50             } ). "</tr>\n";
51 } );
52 ?>
53 <table>
54 <?php echo $ths . $tds ?>
55 </table>
```
Think of it this way. Map reduce can be very good at constructing repeated patterns from arrays.
Example 06 Observations

- Example 06 may be a bit extreme
  - OK to ask, who finds this more readable?
- On the plus side
  - Highly functional limits side-effects.
  - Internal function easy to modify.
- Punchline for CT 310
  1. Study and understand Example 06!
  2. For now avoid loops when you can.
  3. Once you master both styles, then you will be able to make informed choices.
Manipulating Files/Strings

Here is a plain text file:

As a text file, it has three lines
What about if we want to
  Count words?
  Change newlines into HTML newlines?
  Group by words or lines?
What about text from files?

There were three men came out of the west, their fortunes for to try
And these three men made a solemn vow
John Barleycorn must die

Here is the same thing, but with word counts.

There were three men came out of the west, their fortunes for to try (14 Words)
And these three men made a solemn vow (8 Words)
John Barleycorn must die (4 Words)

Here are the words changing new lines and returns to html line breaks.

There were three men came out of the west, their fortunes for to try
And these three men made a solemn vow
John Barleycorn must die
(26 Words Total)
First Part – Split on Newlines

First, note one command reads file.
Now, one command builds an array, one array entry per line.

```php
<?php
    $contents = file_get_contents('ex07src.txt');
    $lines = preg_split("\r\n\", $contents, -1, PREG_SPLIT_NO_EMPTY);
    foreach ($lines as $ln){
        echo "$ln <br/>
";
    }
?>
```

preg_split well worth studying!
Second Part – Word Counts

Ever explode a string?

```php
<?php
    $contents = file_get_contents('ex07src.txt');
    $lines = preg_split("\r\n\", $contents, -1, PREG_SPLIT_NO_EMPTY);
    foreach ($lines as $ln){
        $words = explode(" ", $ln);
        foreach ($words as $w) {
            echo "$w ";
        }
        echo "\n";
    }
    echo "\n";
?>
```

Pay attention to the ease of word counting.

Similar to `preg_split`, but simpler, and better if there is only one delimiter.
Explicitly convert ASCII newlines to line break tags.

```php
<?php
    $contents = file_get_contents('ex07src.txt');
    $words = explode(" ", preg_replace("/\r\n/", " <br> ", $contents));
    foreach ($words as $w) {
        echo "$w ";
    }
    $wc = array_reduce($words, function($s,$n) { return ($n !="<br>") ? $s += $n : $s; },0);
    echo "<br/>(".$wc." Words Total)<br>\n";

Note need to avoid counting <br> tags.