PHP arrays

CIS 1152 Adv Web Dev

Lecture 4

Steve Ruegsegger

Intro

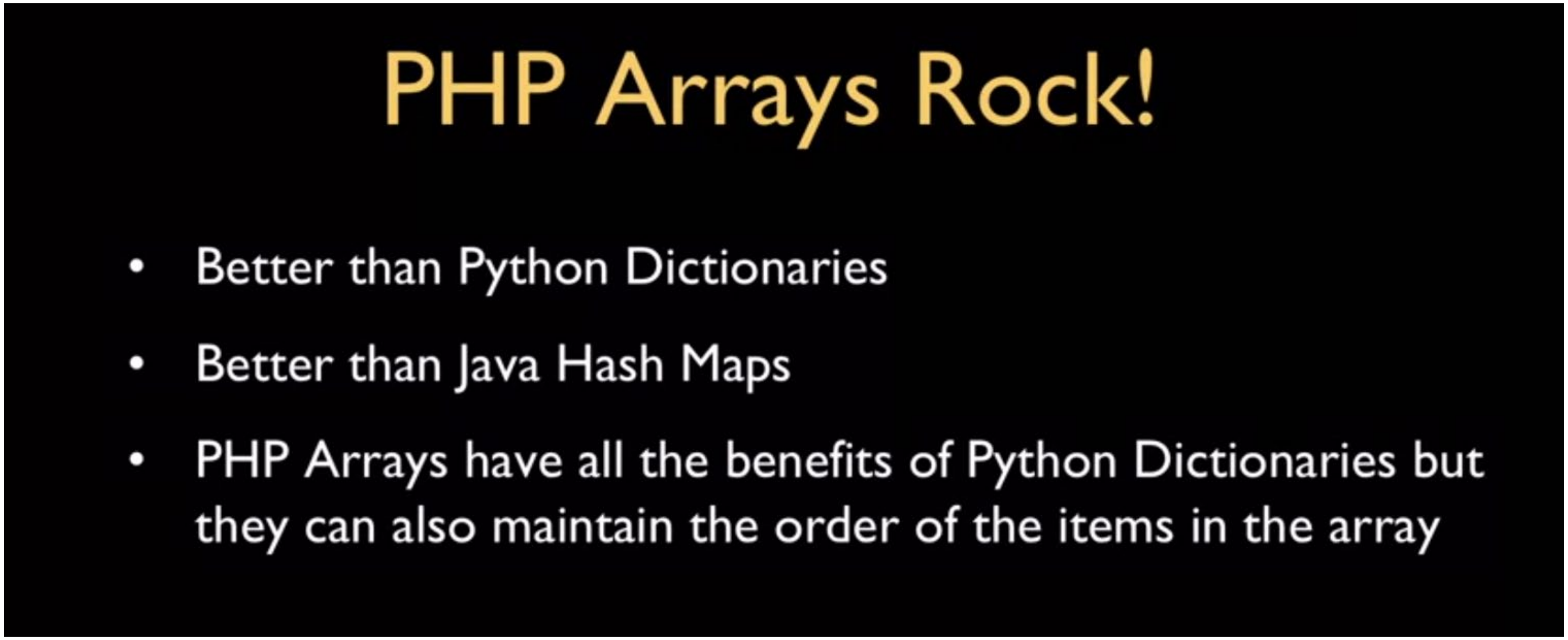
Goal: we need to learn arrays - they are used constantly!

Objectives:

1. Concept
2. Two *types* of arrays
3. Looping over arrays
4. Multi-dimensional arrays
5. Array functions

AKA…

Arrays… Dictionaries… Hash Maps….



PHP Arrays Rock!

- Better than Python Dictionaries
- Better than Java Hash Maps
- PHP Arrays have all the benefits of Python Dictionaries but they can also maintain the order of the items in the array

Why have arrays?

- Let's say we have *a bunch* of something...
- We could make a series of them by putting a number at the end of a common variable base name
- But this is a little cumbersome...

```
$customer1 = "bob"
$customer2 = "joe"
$customer3 = "susie"
...
```

$\underbrace{\qquad\qquad}$

Combine to array $customer

```
$item1 = "bananas"
$item2 = "apples"
$item3 = "pears"
...
```

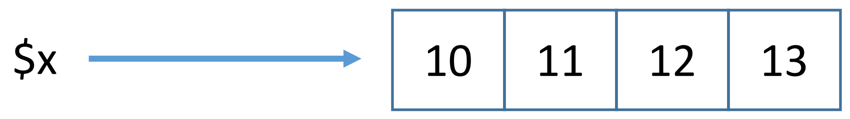
$\underbrace{\qquad\qquad}$

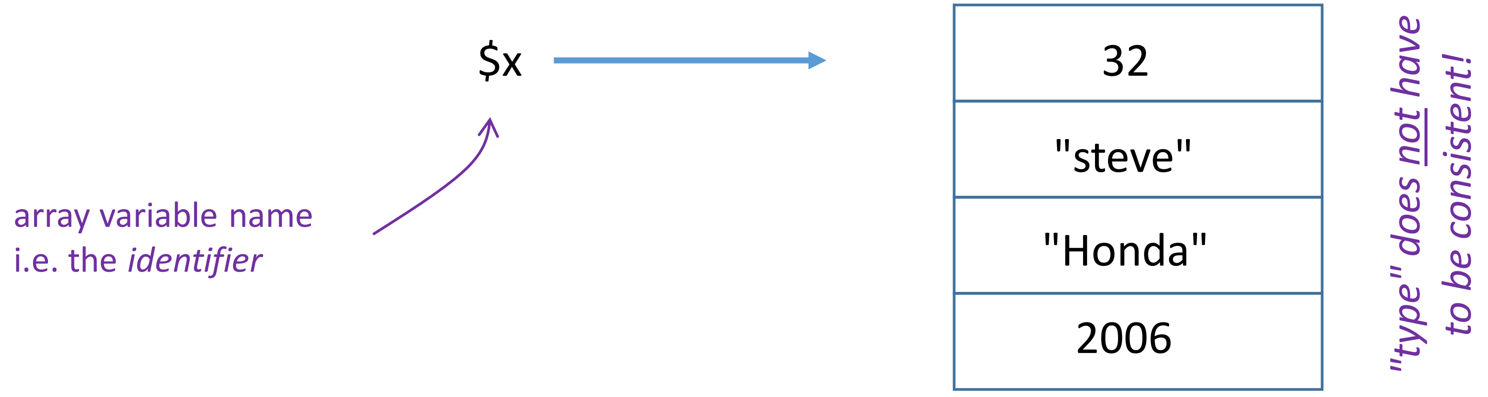
Combine to array $item

Access with individual values with [ ]'s.
$customer[2] = "joe"
$item[3] = "pears"

Conceptualization

- An **array** contains a set of data represented by a single variable name
- The 1 $variable **points** to the ***start*** of a *stack / col / row / list / set* of data





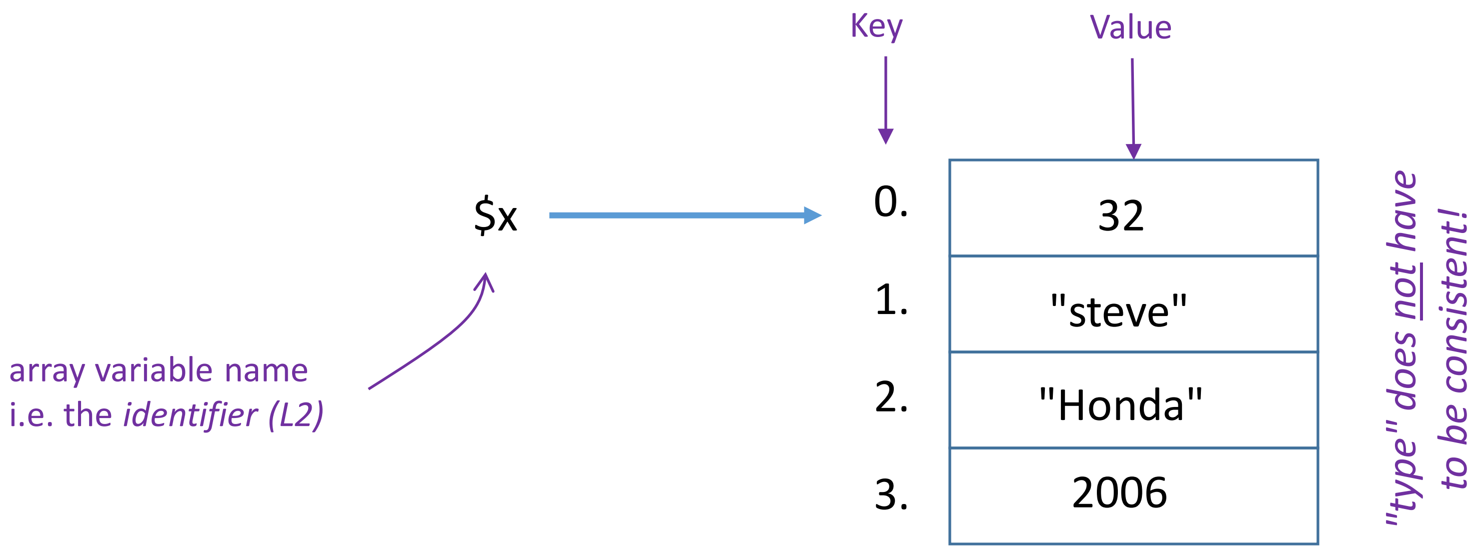
Two types of arrays

"indexed" and "associative"

2 types of arrays

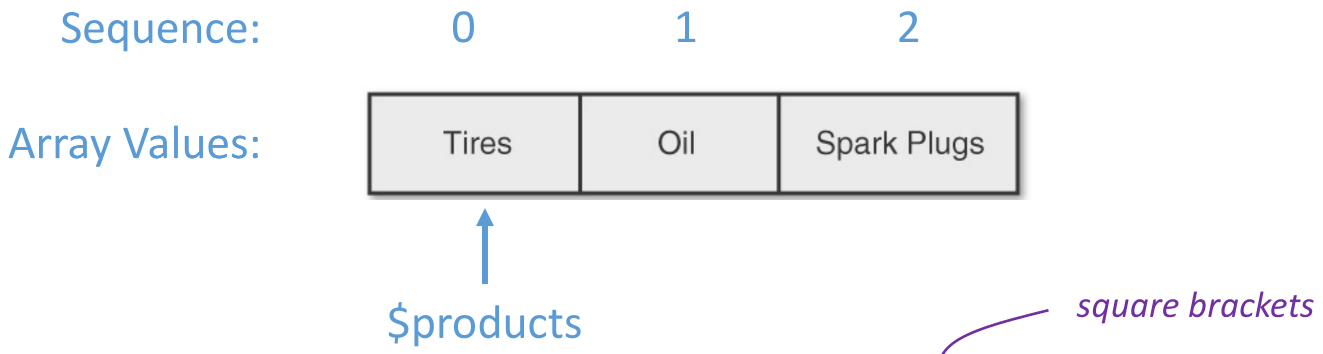
There are *two* ways to think of PHP arrays:

1. **Indexed** – the elements are in a *sequence* and the values are referenced by their index number
2. **Associative** – there is no ordered sequence and the values are referenced by another string '*key*' (key-value *pairs*)


Key
Value
$x
0.
32
1.
"steve"
2.
"Honda"
3.
2006
array variable name
i.e. the *identifier (L2)*
"type" does not have to be consistent!

Type 1. Indexed Arrays

- The array values are *always* in the same **sequential order**
- A.K.A. Sequential Arrays
- We can number the array values' order: 0,1,2,3...
- PHP arrays are *0-based* counting
- Reference an array *element* with the index number
- Use square brackets after the variable name



```
echo "you bought $m of $products[1] <br>";
```

3 ways to define a new indexed array

1. Use `array()` function
2. List all values (comma-separated) in square brackets
3. Add values individually (to end of current list)

```
$products = array( 'Tires', 'Oil', 'Spark Plugs’ );

$products = ['Tires', 'Oil', 'Spark Plugs’];

$products[] = 'Tires';
$products[] = 'Oil’;
$products[] = 'Spark Plugs';
```

Using sequential arrays for set of integers

An array is very helpful for **loops**

- `$ten = range(1,10);`
- `$evens = range(1,100,2);`

-
  ```
  $soup = range('a','m');
  $salad = range('n','z');
  ```

Application: "link" the index numbers

- Applications often have many arrays
- The index number of each array are "correlated" together – to form a complex structure.
- Direct 1:1 link of each index number...
- $cust[0] ordered $quan[0] counts of item $item[0]
- Replace [0] with [1] for next order...

```
$cust[0] = "bob"        $item[0] = "bananas"      $quant[0] = 2;
$cust[1] = "fred"       $item[1] = "pears"        $quant[1] = 2;
$cust[2] = "Susie"      $item[2] = "bananas"      $quant[2] = 5;
```

```
for ($n=0; $n < count($cust) ; $n++) {
    print(" cust=$cust[$n] ordered $quant[$n] of item $item[$n] <br> ";
}
```

Type 2. Associative Arrays

- *a.k.a.* hash (java) or dictionary arrays (python)
- A string *key* points the *value*. (These are key-value pairs.) ☆
- No order to the value. No value is "first" or "last" or "next"
- Only 1 value per key; cannot have 2 values for same key
- Only 1 key to array; cannot have identical keys in one array

Key		**Value (pair)**	**PHP**
Tires	→	2	$order['tires'] = 2;
Oil Filter	→	3	$order['oilfilter'] = 3;
Spark Plugs	→	6	$order['plugs'] = 6;

"Tires" is not first. "Tires" always *points to* "2".

2 ways to make associative arrays

1. Use the `array()` function but put *key-value pairs* with "=>" arrow for each element.
2. Create individually using a string in the square brackets

```
$arrayvar['key'] = value;
```

```
$order = array( 'Tires' => 2,
                'Oil filter' => 3,
                'Spark Plugs' => 6) ;

$order['Tires'] = 2;
$order['Oil filter'] = 3;

# overwrite with new sale price
$order['Oil filter '] = 2;
```

Printing array values

- *Outside* the quotes using comma to separate args to echo

```
echo "You have ", $order['Tires'], " tires in your cart.";
```

- *Warning*: No hashes in double quotes. This will not work:

```
echo "The price for Tires is $prices['Tires']."; # no!
```

- However, curlies {}'s will work *inside* double quotes:

```
echo "You have {$order['Tires']} in your cart.";
echo "You have ${order['Tires']} in your cart.";
```

(both methods above work.)

The array 'key' can be a variable too

```
$states['MI'] = "Michigan";
$states['OH'] = "Ohio";
$states['VT'] = "Vermont";
```

- Why the quotes around the *key*?
  - b/c the key is a string
  - but the key could be the value of a variable! Cool!!!!

```
$mystate = "VT";
echo "Welcome to ", $states[$mystate];
```

Actually...

- Actually, there is only **1 type** of PHP array
- In reality, the sequential array is just an Associative Array with the "key's" being an integer.
- While we don't need to specify a "key" for the *Indexed* (Sequential) array, PHP **knows** how to use the correct sequential key (an integer) by the order in which the array elements are created.

Summary: An Indexed (or Sequential) Array is simply an Associative Array with integers as the "Keys" – and PHP understands that!

Indexed Array	Associative Array
0 → 'Tires'	'MI' → 'Michigan'
1 → 'Oil'	'OH' → 'Ohio'
2 → 'Spark Plugs'	

Looping over arrays

A very logical thing to do!

loop over an array

- 4 methods
- You should know them **all**!

1. The standard `for` loop over the index number
    - protocol: start at 0, strictly < *the length*
    - (obviously, for *indexed* arrays only)

```
for ($m=0; $m<3; $m++) {
   print "item = ", $products[$m], "<br>";
}
```

loop over an array

Your prof likes these "foreach-as" methods for looping.

2. *foreach value* loop:

```
foreach ($array as $var)
```

```
foreach ($products as $current) {
   echo "this item = ", $current, "<br>";
   }
```

3. *foreach key-value* loop:

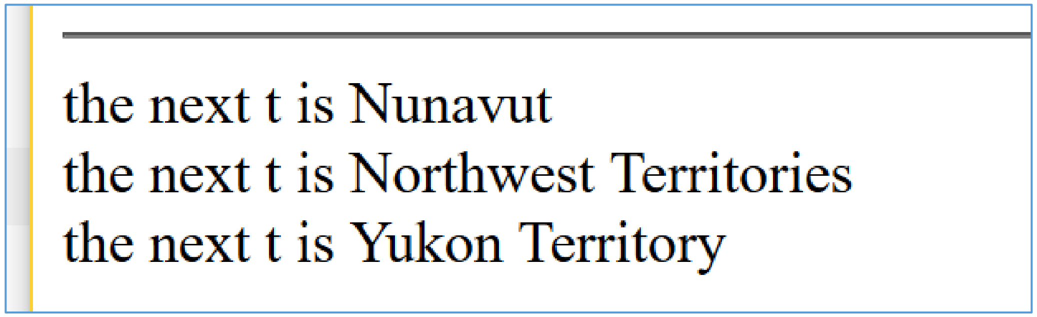
```
foreach ($array as $key => $value)
```

```
foreach ($states as $abb => $state) {
   echo "They abbreviation for $state is $abb<br>";
   }
```

loop over an array

4. current() and next()

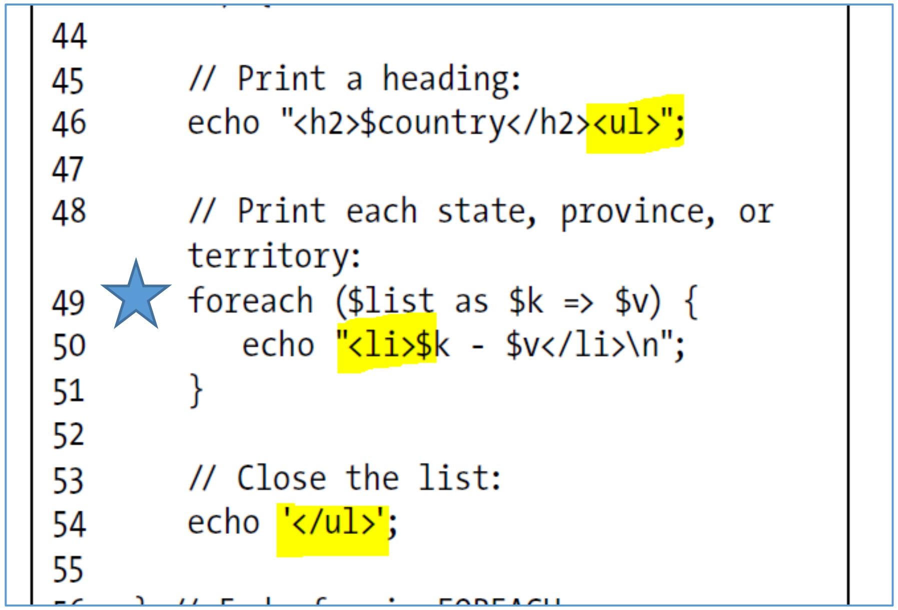
```
echo "<br><hr>";
while($t=current($Territories)) {
  echo "the next t is $t<br>";
  next($Territories);
}
```



```
the next t is Nunavut
the next t is Northwest Territories
the next t is Yukon Territory
```

HTML lists (or tables) from 1D PHP array

1. define the array
2. init the html list or table (**line 46**)   (could be in HTML-land)
3. loop over array (**line 49**)
4. echo an element for each array (**line 50**)
5. close the form element (**line 54**) (could be in HTML-land)



```
44
45      // Print a heading:
46      echo "<h2>$country</h2><ul>";
47
48      // Print each state, province, or
        territory:
49      foreach ($list as $k => $v) {
50          echo "<li>$k - $v</li>\n";
51      }
52
53      // Close the list:
54      echo '</ul>';
55
```

Multi-Dimensional array

2-D

Array of arrays...

Multi-dimensional arrays

- This is really an array of arrays; or nested arrays
- i.e. the values of the outer array, each points to their own array
- It looks like a table.

index	var1	var2	
0	'cell01'	'cell02'	array ←
1	'cell11'	'cell12'	array
2	'cell21'	'cell22'	array

↑
array

name price quantity

```
$products = array( array( 'Tire', 100, 51 )
                   array( 'Spark Plugs', 4, 23)
                   array( 'Oil',12, 62)
                 );
```

Multi-dimensional arrays

- Access values with [row][column] brackets

```
$products[1][2] = 23;
```

```
$products = array( array( 'Tire', 100, 51 )
                   array( 'Spark Plugs', 4, 23)
                   array( 'Oil',12, 62)
                 );
```

# C-style nested looping

Outer loop for each row

```
for ($row = 0; $row < count($product); $row++) {
        for ($col = 0; $col < count($product[$row]); $col++) {
                echo " | ".$product[$row][$col] ;
        }
        echo " | <br>";
}
```

Within each row,
inner loop for each col

Nested looping

- "preferred" (?) PHP-style looping

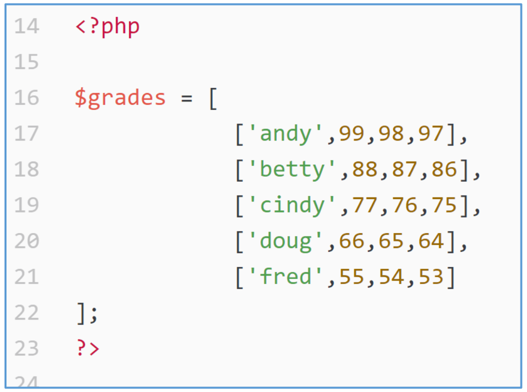
Loops for each row

Within each row,
loop for each col

```
foreach ($products as $row) {
      foreach ($row as $cell) {
            echo " | $cell" ;
      }
      echo " | <br>";
}
```

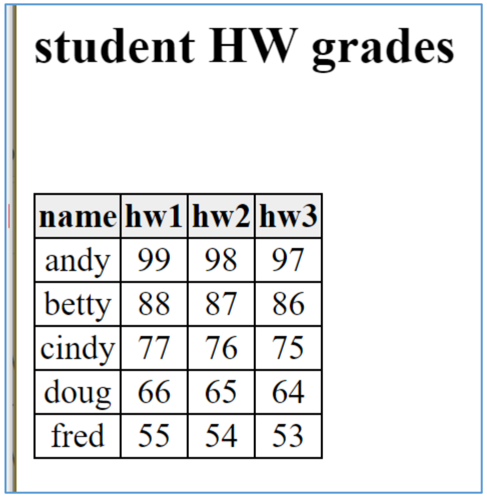
HTML tables from 2D arrays

PHP 2D array



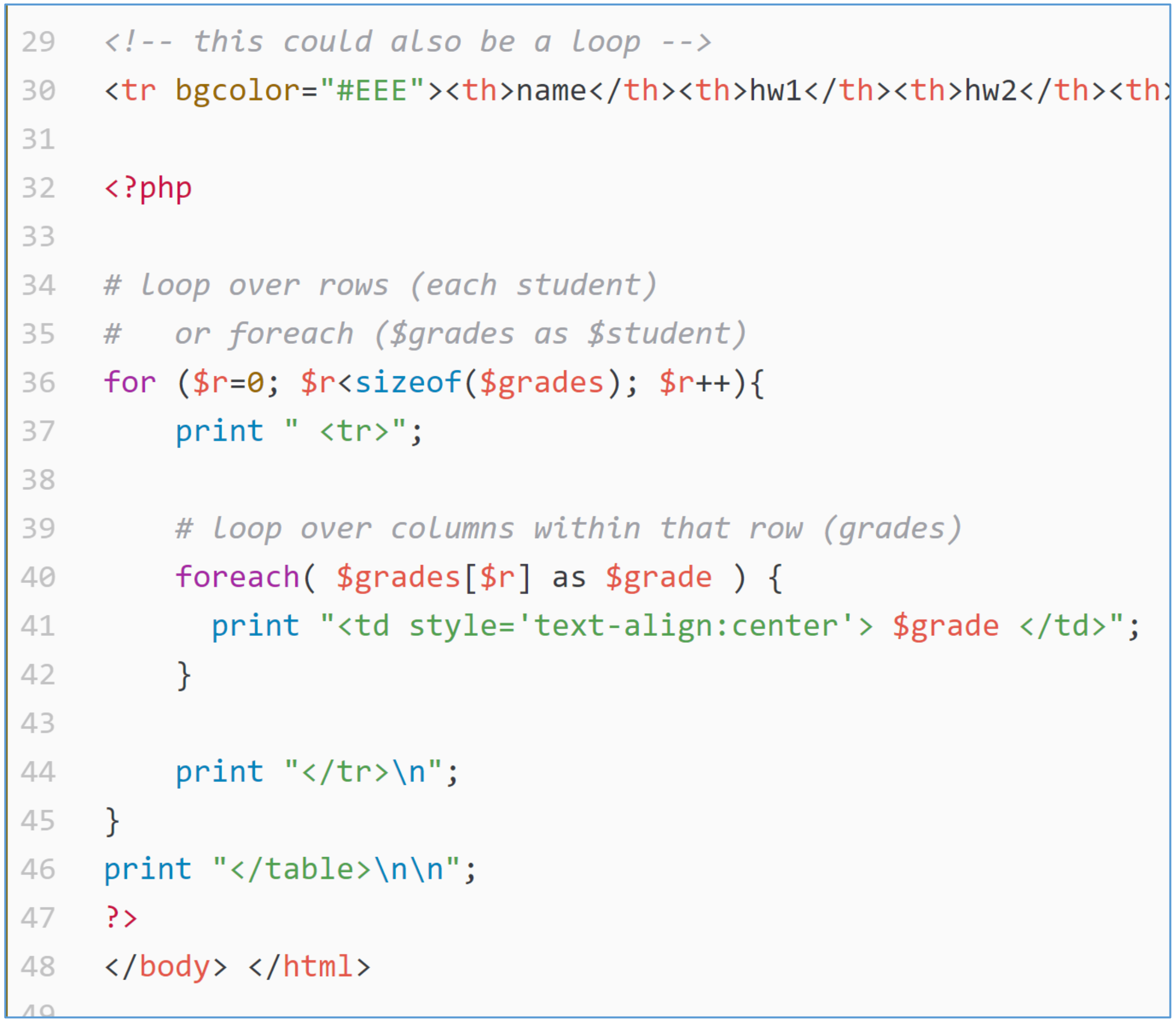
```
<?php

$grades = [
            ['andy',99,98,97],
            ['betty',88,87,86],
            ['cindy',77,76,75],
            ['doug',66,65,64],
            ['fred',55,54,53]
];
?>
```

HTML table



student HW grades

name	hw1	hw2	hw3
andy	99	98	97
betty	88	87	86
cindy	77	76	75
doug	66	65	64
fred	55	54	53



```
<!-- this could also be a loop -->
<tr bgcolor="#EEE"><th>name</th><th>hw1</th><th>hw2</th><th>

<?php

# loop over rows (each student)
#   or foreach ($grades as $student)
for ($r=0; $r<sizeof($grades); $r++){
    print " <tr>";

    # loop over columns within that row (grades)
    foreach( $grades[$r] as $grade ) {
      print "<td style='text-align:center'> $grade </td>";
    }

    print "</tr>\n";
}
print "</table>\n\n";
?>
</body> </html>
```

l04_array2_2D_nestedfor.php

3. Array Functions

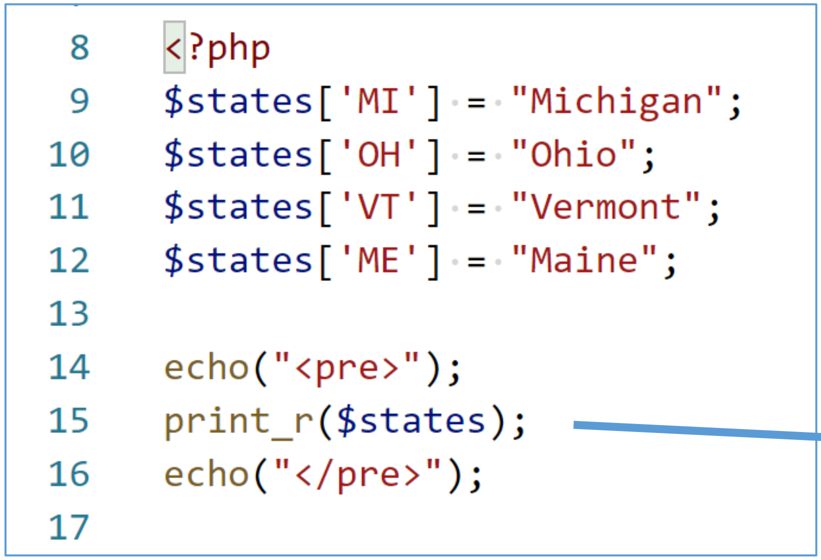
Print that array!

3 different built-in PHP functions!

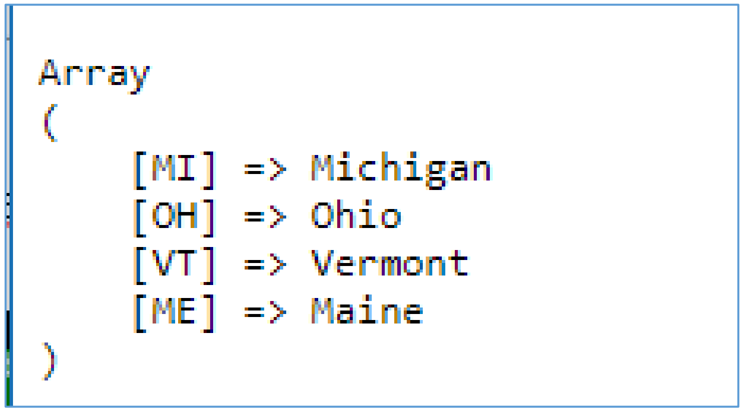
- `print_r()` – displays the index and value of *each element* in an array. *Hint*: wrap in html <pre> .
- `var_dump()` – displays more stuff: the index, value, data type and number of characters in the value
- `var_export()` – similar to `var_dump()` function except it returns valid PHP code

Prof's preference is on the next page!

The prof's go-to debugging code



```php
<?php
$states['MI'] = "Michigan";
$states['OH'] = "Ohio";
$states['VT'] = "Vermont";
$states['ME'] = "Maine";

echo("<pre>");
print_r($states);
echo("</pre>");

```



```
Array
(
    [MI] => Michigan
    [OH] => Ohio
    [VT] => Vermont
    [ME] => Maine
)
```

```
print("<pre>" . print_r($states,true) . "</pre>\n");
```

Sorting Arrays

5 functions

- **`sort`**`()` – **rewrites** and sorts the array based **only** on values. Keys/value pairs are lost! *Don't use this* on associative arrays! This function "throws away the key"

For *associative* arrays, these functions keep key/value relationships:

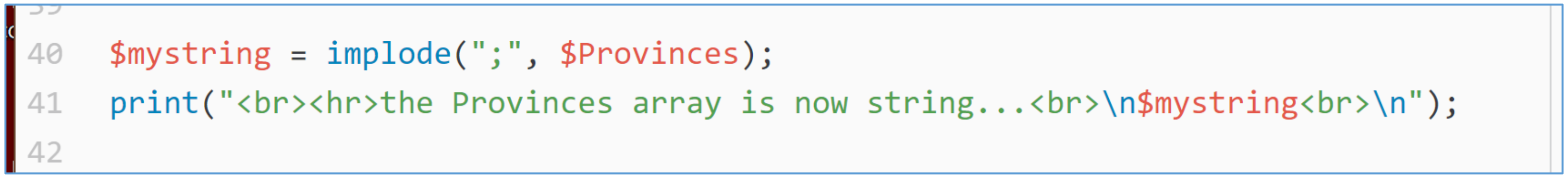
- **`asort`**`()` – rewrites and sorts based on the values
- **`ksort`**`()` – rewrites and sorts based on the keys

```
ksort($states);
foreach ($states as $st) {
        echo "st=$st<br>";
}
```

- 5 functions? There are "reverse" versions of the 2 functions above. Just add an "r" before sort: arsort(), krsort()
- There are even ways to sort with a custom sort algorithm

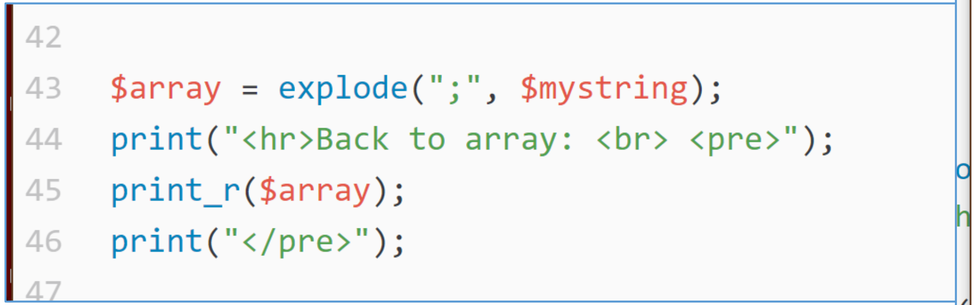
Arrays to strings (and vice versa)

- I use this for **storing** arrays from web forms!
- **implode**: convert the array to a string

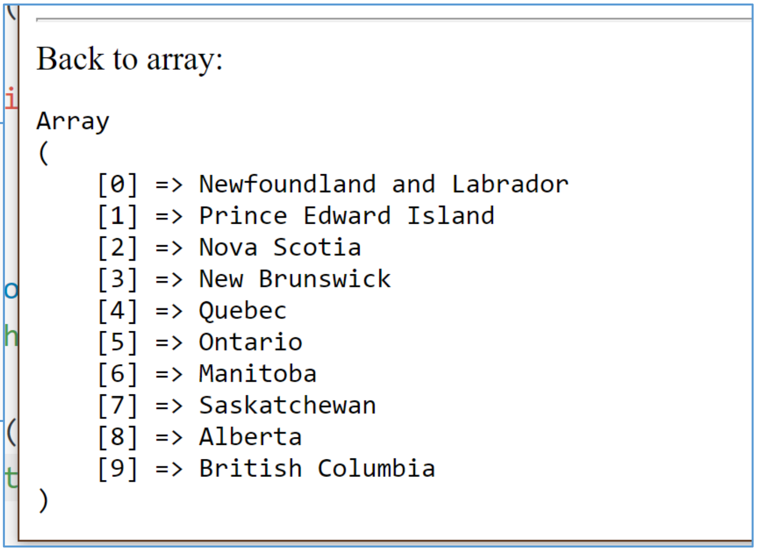


```php
$mystring = implode(";", $Provinces);
print("<br><hr>the Provinces array is now string...<br>\n$mystring<br>\n");
```

- **explode**: convert a delimited string to an array



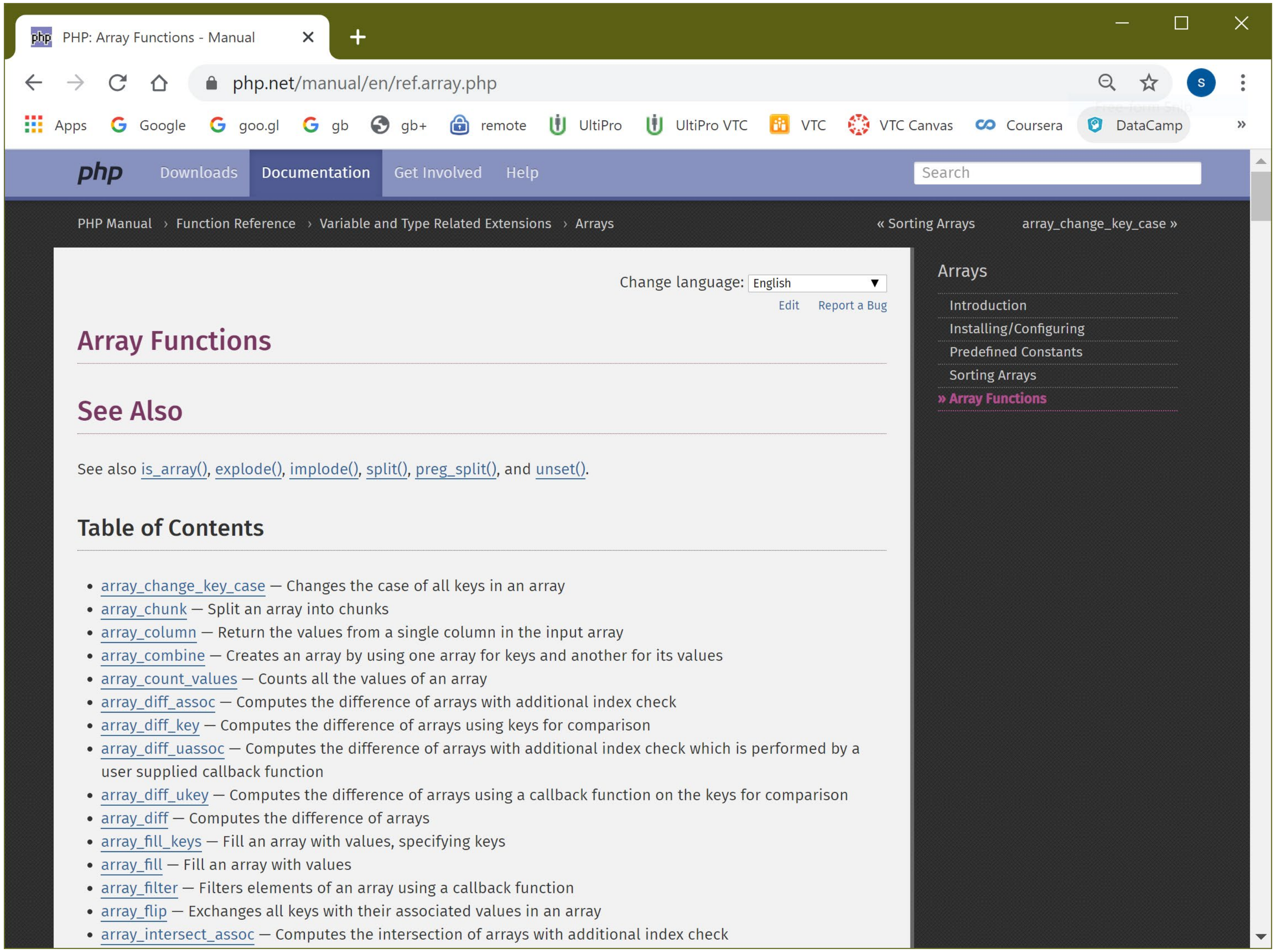
```php
$array = explode(";", $mystring);
print("<hr>Back to array: <br> <pre>");
print_r($array);
print("</pre>");
```



Back to array:

```
Array
(
    [0] => Newfoundland and Labrador
    [1] => Prince Edward Island
    [2] => Nova Scotia
    [3] => New Brunswick
    [4] => Quebec
    [5] => Ontario
    [6] => Manitoba
    [7] => Saskatchewan
    [8] => Alberta
    [9] => British Columbia
)
```

Many array functions


PHP: Array Functions - Manual
php.net/manual/en/ref.array.php
Apps Google goo.gl gb gb+ remote UltiPro UltiPro VTC VTC VTC Canvas Coursera DataCamp
php Downloads Documentation Get Involved Help
Search
PHP Manual › Function Reference › Variable and Type Related Extensions › Arrays
« Sorting Arrays
array_change_key_case »
Change language: English
Edit Report a Bug
Arrays
Introduction
Installing/Configuring
Predefined Constants
Sorting Arrays
» Array Functions

Array Functions

See Also

See also is_array(), explode(), implode(), split(), preg_split(), and unset().

Table of Contents

- array_change_key_case — Changes the case of all keys in an array
- array_chunk — Split an array into chunks
- array_column — Return the values from a single column in the input array
- array_combine — Creates an array by using one array for keys and another for its values
- array_count_values — Counts all the values of an array
- array_diff_assoc — Computes the difference of arrays with additional index check
- array_diff_key — Computes the difference of arrays using keys for comparison
- array_diff_uassoc — Computes the difference of arrays with additional index check which is performed by a user supplied callback function
- array_diff_ukey — Computes the difference of arrays using a callback function on the keys for comparison
- array_diff — Computes the difference of arrays
- array_fill_keys — Fill an array with values, specifying keys
- array_fill — Fill an array with values
- array_filter — Filters elements of an array using a callback function
- array_flip — Exchanges all keys with their associated values in an array
- array_intersect_assoc — Computes the intersection of arrays with additional index check

Non-existing key

- What if you ask for a *key* that doesn't exist → fatal error (*yuck*)
- So, if you are *not sure* if a key exists, you need to check with a Boolean function returns T or F
- This is common for web-forms, so we'll re-visit this topic later

```
if (array_key_exists('key',$array) {
        # exists
} else {
        # does not exist
}
```

Prof Preference:

- We use isset() to check without error, and return a default if F
- We can use the ternary shortcut operator as a shortcut!

★

```
$value = isset($array['key']) ? $array['key'] : "n/a default" ;
```

Lab 3 this week

We will use PHP arrays ...