ARRAYS

DR. ANDREW BESMER

- Arrays
- Sequential Arrays
- Associative Arrays
- Multi-dimensional Arrays
- Additional Users
- Additional Functions

ARRAYS

ABOUT ARRAYS

- What if you needed to average all of the grades for a student in a class? One possibility is:

```php
<?php

$grade1 = 88;
$grade2 = 92;
$grade3 = 75;
...
...
$gradeN = 97;
```

  - What sort of issues would this have?

ABOUT ARRAYS

- Arrays can hold multiple values
  - Each value is called an **element**
- Lend themselves well to looping structures we have learned
- It’s not required to know how many you need to store at the time of development

ABOUT ARRAYS

- PHP arrays are not really arrays, they are much more flexible
- PHP arrays allow values of different types to be stored
  - Not limited to only `integer` or only `float`
  - Typed languages generally do not allow this
  - Underneath arrays are actually hash maps

SEQUENTIAL ARRAYS

SEQUENTIAL ARRAYS

- Lets look at storing a students grades in an array
- Observe
    - Structure
    - Indexing - zero based
    - Retrieval
    - Properties

CREATING ARRAYS

- Use `array()` to create arrays
  - Pass your values each separated by a comma

```php
<?php

$grades = array(88, 92, 75, 97);

var_dump($grades);
```

ACCESSING ARRAY

- Subscripts `[]`
  - integer - `$grades[0]`
  - string - `$grades["0"]`
- Casting
  - booleans - `true` to `1` and `false` to `0`
  - floats - `5.4` to `5`
  - strings - Depends!

DELETING ARRAY ELEMENT

- `unset($grades[0])` - Removes item from an array
- `unset($grades)` - Removes whole array

CREATING ARRAYS

- Optionally use `[ ]` for assigning one at a time

```php
<?php

$grades[] = 88;
$grades[] = 92;
$grades[] = 75;
$grades[] = 97;

var_dump($grades);
```

ASSOCIATIVE ARRAYS

ASSOCIATIVE

- Optionally create by specifying a key value pair with the `=>` operator

```php
<?php

$grades = array("test1" => 88, "hw1" => 92, "hw2" => 75, "midterm" => 97);

var_dump($grades);
```

ACCESSING ARRAY

- Access using string `$grades["test1"]`

ACCESSING ARRAY

- Can but should not use regular for loop
  - `count()` function and access in other languages
  - Instead use `foreach`

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

foreach($myArray as $key => $value)
{
    echo $key . " " . $value;
}
```

ACCESSING ARRAY

- Iteration
  - Order added not indexed
  - Linked list

MULTI-DIMENSIONAL ARRAYS

MULTI-DIMENSIONAL ARRAYS

- Arrays within arrays

```
$grades = array("Andrew" => array("test1" => 88, "hw1" => 92, "hw2" => 75, "mi

var_dump($grades);
```

- How can you access my test1 score?

ADDITIONAL USERS

STACK

- Stacks are LIFO
  - `array_push($myArray, $value)`
  - `array_pop($myArray)`

QUEUE

- Queues are FIFO
  - `array_shift($myArray)`
  - `array_unshift($myArray, $value)`

SORTING

- `sort()` numerically and alphabetically ascending
  - New sorted array starts at 0
- `rsort()` numerically and alphabetically descending

- `asort()` numerically and alphabetically ascending keeping index key

ADDITIONAL FUNCTIONS

FUNCTIONS

- `isset($grades["Andrew"])` - check if an element exists
- `max($grades["Andrew"])` - get max element
- `min($grades["Andrew"])` - get min element
- `sum($grades["Andrew"])` - sum all elements
- `count($grades["Andrew"]")` - count of elements