Associative arrays

- Associative arrays map a *key* to a *value*
  - Keys and values can be different data types

```
"name" => "Emmett"
"arms" => 2
42 => "The answer"
```

- Associative arrays can be implemented in many ways
  - Parallel array
  - Array of key-value structs
  - Hash table
  - Vector

Ordered maps

- PHP implements associative arrays as *ordered maps*
  - The key-value pairs are stored in a particular sequence
- Standard array-subscript notation is typically used
  - With keys instead of integer indeces
  - Supports both assignment and retrieval

```
$a ["key_one"] = 72;
print $a["key_one"];
```

  - Setting the value for a non-existent key creates the key-value pair

```
$a["newkey"] = 7;
```

  - Retrieving the value for a non-existent key returns NULL

```
($a["newkey"] == NULL)  # is TRUE
```

Iterating over associative arrays

- Unless the keys are contiguous integers (0,1,2,3…), a standard for-loop doesn't make sense
  - It will just return NULL for all the non-existent keys

```
for( $i=0; $i<count($a); $i++ )
{
   print "<h1>$a[$i]</h1>\n";
}
```

Iterating over associative arrays

- Instead, like many scripting languages, PHP has a convenient *foreach* loop
  - Iterates over the array values in order
  - Loop syntax specifies:
    - The array to iterate over ($myarr)
    - A variable name to *bind* each successive value to ($a)

```
foreach( $myarr as $a )
{
   print "<h1>$a</h1>\n";
}
```

Iterating over associative arrays

- Foreach can also iterate over the keys and values
  - Syntax specifies variables to bind each key and value

```
foreach( $myarr as $k => $v )
{
   print "<li>Key: $k, Value: $v\n";
}
```

Simulating indexed arrays

- Associative arrays can always be used like indexed arrays
  - Simply use contiguous integers as keys
- PHP provides shortcut appending with an empty subscript

```
$arr = array( “a”, “b”, “c” );    # keys 0, 1, 2
$arr[] = “d”;                     # “d” has key 3
```

Keys and values

- Retrieve the keys from an array

```
array_keys( $a );
```

  - The keys are returned as an array with indeces 0,1,2…

- Retrieve the values from an array

```
array_values( $a );
```

  - The values are returned as an array with indeces 0,1,2…

Removing array items

- Setting a key to the value NULL…

```
$a["name"] = NULL;
```

  - Is kind of like removing it:

```
print $a["name"];  # prints NULL
```

  - But not really:

```
array_keys( $a );  # still one of the keys
count( $a );       # still counted
```

- The unset() function deletes variables

```
unset( $myVar );
```

  - Including key-value pairs in an array

```
unset( $a["name"] );
```