Using Arrays in PHP

Lecture 5

Introduction

- Array is a collection of variables which can be accessed through an index or key.
- The array's elements can be from different data types.

Introduction

- There are two types of arrays in PHP:
  - **Indexed arrays:** using numbers in their index.
  - **Associative arrays:** using strings in their index.
- Arrays can be:
  - **One dimensional** arrays
  - **Multidimensional** arrays

Table of Content

Creating Indexed Arrays

- By using `array()`

  Syntax:

  ```
  $arrayName = array();
  ```

  Or

  ```
  $arrayName = array (v1, v2, v3);
  ```

  Example:

  ```
  $colors = array("Red","Blue");
  ```

- A complete an initialized array will be created.

Creating Indexed Arrays cont.

- By using `[]`with index

  Syntax:

  ```
  $var[n] = v1;
  ```

  Example:

  ```
  $colors[0]= "Red";
  $colors[1]= "Blue";
  ```

- You can specify the index of your choice and assign it with a value.

Creating Indexed Arrays cont.

- By using `[]`without index

  Syntax:

  ```
  $var[] = v;
  ```

  Example:

  ```
  $colors[]= “Red”;
  $colors[]= “Blue”;
  ```

- Each time [] is used, the element is added at the end of the array.

Creating Associate Arrays

- By using `array()`

  Syntax:

  ```
  $arrayName = array(key=>value);
  ```

  Example:

  ```
  $days = array(“Thu”=>”Thursday”,
          “Fri”=>“Friday”);
  ```

- A complete an initialized array will be created. And each key will referred to the value after =>.

Creating Associate Arrays cont.

- By using `[]`with index

  Syntax:

  ```
  $var[“key”] = v1;
  ```

  Example:

  ```
  $days[“Thu”]= “Thursday”;
  $days[“Fri”]= “Friday”;
  ```

- It is useful way when you want to add elements after the initialization.

Creating Multidimensional Arrays

- It is like creating a one-dimensional array. But instead of using one value, we replace it by another array definition using `array()`.

  Associate Syntax:

  ```
  $arrayName = array(key=>array(v1,v2),
               key2=>array(v1,v2));
  ```

  Indexed Syntax:

  ```
  $arrayName = array(array(v1,v2),
               array(v1,v2));
  ```

Creating Multi-dim Arrays cont.

- Example

```
$Student=array(
        “ali”=>array(“42600151”,
                    “24”,”3.5” ),
        “ola”=>array(“42601115”),
                    “21”,”2.9”)
            );
```


Accessing Arrays Elements

- By using the element’s key or index enclosed by [].

Example:

```
echo $colors[0];
echo “<br>”.$days[“fri”];
echo “<br>”.$student[“ali”][2];
```

Outputting Array Elements

- By using `print_r()` you can print the array keys and elements.

  Syntax:

```
print_r($arrayName);
```

  Example:

```
print_r($days);
//   Array
//   {
//        [thu] =>   “Thursday”
//        [fri] =>   “Friday”
//   }
```


Outputting Array Elements cont.

- By using `var_export()` you can print the array keys and elements.

  Syntax:

```
var_export($arrayName);
```

  Example:

```
var_export($days);
//  array(
//       “thu”    =>  “Thursday”,
//       “fri”    =>  “Friday”
//  )
```

Outputting Array Elements cont.

- By using `var_dump()` you can print the array keys and elements.

  Syntax:

```
var_dump($arrayName);
```

  Example:

```
var_dump($days);
//   array(2)   {
//      ["thu"] =>string(8) "Thursday"
//      ["fri"] =>string(6) "Friday"
//   }
```


Adding New Elements to an Array

- **Adding to associate array:**

  Use the [] with or without an index or key and assign the desired value.

  Example:

```
$days["Sat"]="Saturday";
```

Adding New Elements cont.

- **Adding at the end of array:**

  Use the [] with or without assign the desired value.

  Example:

```
$colors[]="Black";
```


Adding New Elements cont.

- **Adding at the end of array:**
  - Use the [] with or without assign the desired value.
    Example:
    `$colors[]="Black";`
  - Use `array_push($arrayName, v1, v2):`
    Example:
    `array_push($color,"Green","White");`

Adding New Elements cont.

- **Adding at the beginning of array:**
  - Use the array_unshift().
    Syntax:
    ```
    int array_unshift($arr, $v1, $v2,..);
    ```
    It will return the number of element in the new array.

    Example:
    ```
    array_unshift($days, "Wed"=>"Wednesday");
    ```


Merging Arrays

- By using `array_merge()`
- Syntax:
  ```
  $newArray=array_merge($ar1, $ar2);
  ```

  Example:
  ```
  $array=array_merge($colors, $days);
  ```

Merging Arrays cont.

- By using `+ operator`

  Syntax:

  ```
  $newArray= $ar1+$ar2;
  ```

  Example:

  ```
  $array= $colors + $days;
  ```

- Beware of `+ operator` with indexed arrays because it works as a union to the elements according to their indexes. So if there are two element with the same index one will be dropped.

Comparing Arrays

- $arr1 == $arr2
  is true if and only if they are equal in length, keys and elements but not order.
- $arr1 === $arr2
  is true if and only if they are equal in length, keys, elements and order.
- $arr1 > $arr2
- is true if and only if $arr1 has more element that arr2. (i.e. comparing their length).

Counting Array Elements

- By using count()

Syntax:

```
int count($arrayName);
```

Example:

```
echo count($days);
//4
```


Traversing Arrays

- By using `foreach()` (for associate & indexed).

  Syntax:

```
foreach($arrayName, $key=>$value)
{           }
```

- It will start from the first element to the last. In each iteration the current key will be saved in `$key` and its corresponding value will be saved in `$value`.

Traversing Arrays cont.

- By using `for` and `count()` (for indexed).
  Syntax:

```
for($i=0; i<count($array);$i++)
{            }
```

- The current index is `$i` and the current value is `$array[i]`

Example

- Build a form for a shopping cart. Whenever the user submit his/her order, a confirmation page will appear with the reciete details. Make sure that the total price had been spelled out.
  - All this will be done on the same page.
  - Build spell_out function to spell out any given number into words.