

---

# Rapid Instruction Unit

## **PHP: Arrays**

# **PHP: Arrays**

---

## **Part 1 – Array Definition**

---

# PHP: Arrays

## Part 1 – Array Definition

---

Arrays are a type of variable for storing multiple values

Arrays provide an addressable structure to store and retrieve values

Arrays are an aggregate or compound data type

Arrays may be named for identification and referencing

*\$myArray    \$\_thislist    \$a\_descriptive\_name    \$my2ndArray*

Array names adhere to the same naming rules applied to all PHP variables

*valid PHP variable names...*

- *begin with \$*
  - *followed by a letter or underscore*
  - *followed by any number of letters, numbers, or underscores*
-

# PHP: Arrays

## Part 1 – Array Definition

---

Arrays represent a single row of data, like a list

*{1,2,3,4,5,6...}*    *{a,b,c,d,e,f...}*    *{one,two,three,four...}*    *{apples,pears,bananas...}*

Arrays can store **scalar** data values *as well as other arrays*

*{apples,pears,bananas}*

*{fruit{apples,pears,bananas},meat{filet,turkey,chicken}...}*

Arrays use **keys** to identify the location of each stored value

*\$name\_of\_array[key]*

Array keys may be **indexed** (*numeric*) or **associative** (*named*)

*\$myshoppinglist[0]*

*\$demographics[population]*

---

**scalar**: (*skālār*, *-lār*), adj.<programming> A data type representing a single value (e.g. a number or Boolean), as opposed to an aggregate data type that has many elements or values. A string is regarded as a scalar in most languages (e.g. C, Perl, PHP...).

# PHP: Arrays

## Part 1 – Array Definition

---

Array key indexes start at 0 if assigned automatically

name	→	<i>\$myshoppinglist</i>			
key	→	0	1	2	3
value	→	<i>apples</i>	<i>bananas</i>	<i>milk</i>	<i>orange juice</i>

The example array, *\$myshoppinglist[]*, contains 4 values:

The value “*apples*” is stored in *\$myshoppinglist[0]*

The value “*bananas*” is stored in *\$myshoppinglist[1]*

The value “*milk*” is stored in *\$myshoppinglist[2]*

The value “*orange juice*” is stored in *\$myshoppinglist[3]*

---

# PHP: Arrays

## Part 1 – Array Definition

---

Arrays using numeric keys have a **numbered index**

name    **→**    *\$my\_numbered\_shopping\_list*

key     **→**

value  **→**

0	1	2	3
<i>apples</i>	<i>tomatoes</i>	<i>milk</i>	<i>london broil</i>

Arrays using named keys have an **associative index**

name    **→**    *\$my\_named\_shopping\_list*

key     **→**

value  **→**

<i>fruit</i>	<i>vegetable</i>	<i>beverage</i>	<i>meat</i>
<i>apples</i>	<i>tomatoes</i>	<i>milk</i>	<i>london broil</i>

Array keys represent a specific location for a value stored in an array

In the first array above, the value “*apples*” is stored in *\$myshoppinglist[0]*

In the second array above, the value “*apples*” is stored in *\$myshoppinglist[fruit]*

---

# PHP: Arrays

## Part 1 – Array Definition

---

Arrays keys must be unique

### *Associative keys*

name → *\$myshoppinglist*

key →

value →

<i>fruit</i>	<i>vegetable</i>	<del><i>vegetable</i></del>	<i>meat</i>
<i>apples</i>	<i>tomatoes</i>	<i>lettuce</i>	<i>london broil</i>

### *Indexed keys*

name → *\$myshoppinglist*

key →

value →

<i>0</i>	<i>1</i>	<i>2</i>	<del><i>2</i></del>
<i>apples</i>	<i>tomatoes</i>	<i>lettuce</i>	<i>london broil</i>

---

# PHP: Arrays

## Part 1 – Array Definition

---

Array keys and corresponding values may be referred to as **key/value pairs**

name    **→**    *\$myshoppinglist*

<b>key</b> <b>→</b>	<i>fruit</i>	<i>vegetable</i>	<i>beverage</i>	<i>meat</i>
<b>value</b> <b>→</b>	<i>apples</i>	<i>tomatoes</i>	<i>milk</i>	<i>london broil</i>

The key/value pairs for the array *\$myshoppinglist* are

*fruit,apples*

*vegetable,tomatoes*

*beverage,milk*

*meat,london broil*

---

Arrays may be loaded with values using simple assignments

1. Assign values using “name-bracket” notation. An **automatic index** is created, and data is appended with each statement:

```
$myshoppinglist[] = 'apples';
```

```
$myshoppinglist[] = 'tomatoes';
```

```
$myshoppinglist[] = 'milk';
```

results in ‘*apples*’ stored at *\$myshoppinglist[0]*, ‘*tomatoes*’ at *\$myshoppinglist[1]*...

2. Assign values directly to an **arbitrary index** using “name-bracket” notation, data is appended with each statement:

```
$myshoppinglist[12] = 'apples';
```

```
$myshoppinglist[13] = 'tomatoes';
```

```
$myshoppinglist[21] = 'milk';
```

results in an array named *\$myshoppinglist* with 3 keys (*12,13,21*) and the 3 values (*'apples', 'tomatoes', 'milk'*) assigned to each respectively

---

Arrays may be loaded with values using the `array()` function

1. Assign values **without keys**, resulting in an **automatic index**:

```
$myshoppinglist = array ('apples', 'tomatoes', 'milk', 'london broil');
```

results in 'apples' stored at `$myshoppinglist[0]`, 'tomatoes' at `$myshoppinglist[1]`, 'milk' at `$myshoppinglist[2]`, 'london broil' at `$myshoppinglist[3]`

2. Assign values **with numeric keys**, using the **key=>value** notation to produce an **arbitrary, numeric index**:

```
$myshoppinglist = array (12=>'apples', 13=>'tomatoes', 21=>'milk',  
43=>'london broil');
```

results in 'apples' stored at `$myshoppinglist[12]`, 'tomatoes' at `$myshoppinglist[13]`, 'milk' at `$myshoppinglist[21]`, 'london broil' at `$myshoppinglist[43]`

---

Arrays may be loaded with values using the `array()` function

3. Assign values **with named keys**, using the **key=>value** notation to produce an **associative index**:

```
$myshoppinglist = array ('fruit'=>'apples', 'vegetable'=>'tomatoes',  
                        'beverage'=>'milk', 'meat'=>'london broil');
```

The above statement results in an array with 4 named keys (`'fruit'`, `'vegetables'`, `'beverage'`, `'meat'`) and the 4 values (`'apples'`, `'tomatoes'`, `'milk'`, `'london broil'`) assigned to each respectively.

The value `"apples"` is stored in `$myshoppinglist['fruit']`

The value `"tomatoes"` is stored in `$myshoppinglist['vegetables']`

The value `"milk"` is stored in `$myshoppinglist['beverage']`

The value `"london broil"` is stored in `$myshoppinglist['meat']`

---

# PHP: Arrays

## Part 1 – Array Definition

---

Arrays may be loaded with values using the `array()` function

4. Assign values **with named keys**, using the **key=>value** notation to produce an **associative index**, using scalar values **and arrays**:

### Scalar Assignments:

```
$fruit = array ('apples'=>'1 bag', 'pears'=>'2 bags', 'grapes'=>'3 bunches') ;  
$drinks = array ('orange juice'=>'1 gal.', 'apple juice'=>'2 qts.', 'kool-aid'=>'1 lb.') ;  
$meat = array ('filet mignon'=>'2 lbs.', 'ground turkey'=>'4 lbs.', 'chicken'=>'4 lbs.') ;
```

### Array Assignments:

```
$myshoppinglist = array('fruit'=>$fruit, 'drinks'=>$drinks, 'meat'=>$meat);
```

The last statement above results in the array named `$myshoppinglist` with 3 named keys (`'fruit'`, `'drinks'`, `'meat'`) and the contents of 3 arrays (`$fruit`, `$drinks`, `$meat`) assigned to each respectively.

The value “1 bag” is stored in `$myshoppinglist['fruit']['apples']...`

The value “1 lb.” is stored in `$myshoppinglist['drinks']['kool-aid']...`

The value “4 lbs.” is stored in `$myshoppinglist['meat']['ground turkey']...`

---

# PHP: Arrays

---

**End of Part 1**

---

# **PHP: Arrays**

---

## **Part 2 – Array Manipulation**

---

# PHP: Arrays

# References

---

*PHP Manual*. (2003). Bakken, S.S. & Schmid, E., (Eds.), PHP Documentation Group. <http://www.php.net/manual/en/>

Lerdorf, R. (2002). *Programming PHP*. Sebastopol: O'Reilly.

Wise, L. (2003). *PHP Basics*.

<http://www.csse.monash.edu.au/courseware/cse2030/2003/phpbasics.ppt>

---