CREATE AN ASSOCIATIVE ARRAY

You can create an associative array, which allows you to use string values instead of index numbers to identify the elements in the array. These string values are often referred to as keys.

Creating an associative array is similar to creating a simple array. You create a variable to store the array and then use the `array` keyword to assign the elements. When assigning an element to an associative array, however, you must first type the key. The key is enclosed in quotation marks and should describe its associated value in the array. The key is followed by the `=>` operator and the value of the element. Each key-value pair in an associative array is separated by a comma.

It is good programming practice to avoid using spaces and non-alphanumeric characters, such as ? and &, in the keys of an associative array. You should also try to use short but meaningful keys. Arrays containing many elements with very long keys require more memory, which makes the script less efficient.

To access an element in an associative array, you specify the name of the array followed by the key enclosed in brackets, such as $user[name]. If the key contains spaces, it must be enclosed in quotation marks, such as $user["last name"].

You should not access an element with a key that is enclosed in quotation marks within another string, such as print "Welcome $user["first name"]", since an error will be generated. In such cases, it is better to access the element outside the string and use the concatenation operator (.) to combine the strings, such as print "Welcome " . $user["first name"].

Apply it

You can use the `foreach` statement to work with each key-value pair in an associative array. You specify another variable to hold the value of each key after the `as` keyword in the `foreach` statement. This variable is followed by the `=>` operator and the variable that will hold the value of each element.

TYPE THIS:

```
$user = array ("name" => "Tom",
               "age" => 24,
               "gender" => "male");
foreach ($user as $key => $value)
{
     print "$key: $value<br>";
}
```

RESULT:

name: Tom
age: 24
gender: male

To change the value of an element in an associative array, you specify the name of the array followed by the key of the element you want to change. You can then use the assignment operator (=) to assign a new value.

Example:

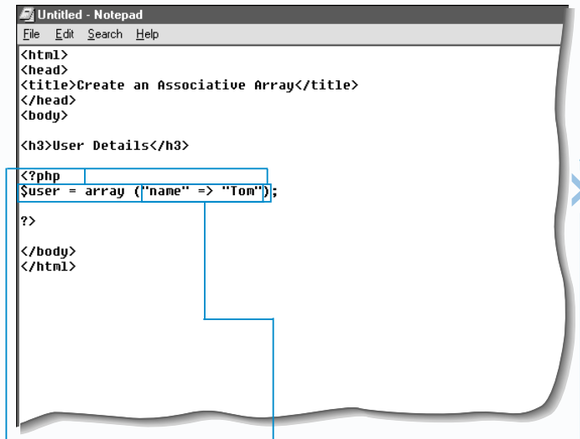
```
$user = array ("name" => "Tom",
               "age" => 24,
               "gender" => "male");
$user[age] = 29;
```

You can later add an element to an associative array. To do so, you specify the name of the array followed by the key of the new element. You can then assign the new value.

Example:

```
$user[status] = "single";
```

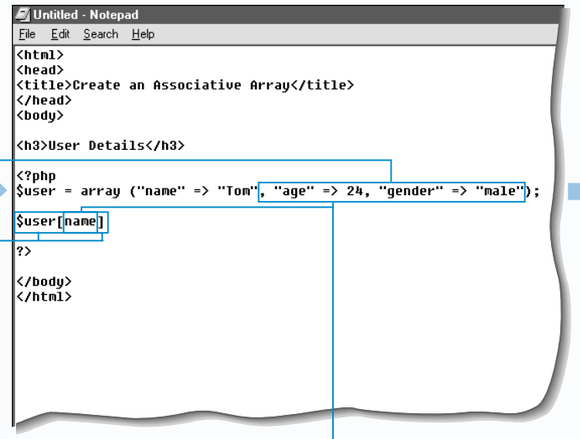
CREATE AN ASSOCIATIVE ARRAY



1 Type a name for the associative array you want to create followed by **= array ()**.

2 To add an element to the array, type a key followed by **=>** and the value you want to assign to the element.

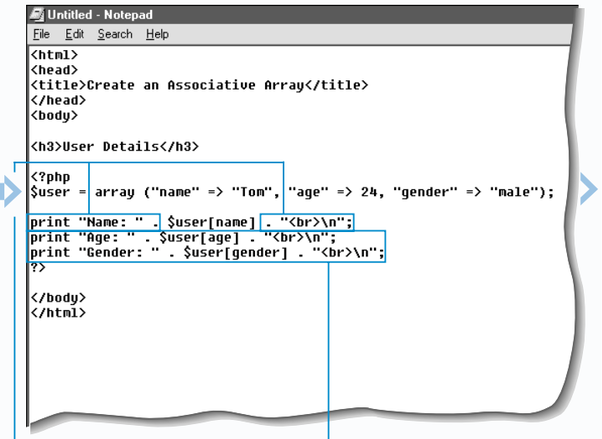
- The key and any string values must be enclosed in quotation marks.



3 Repeat step 2 for each element you want to add to the associative array. Separate each key-value pair with a comma.

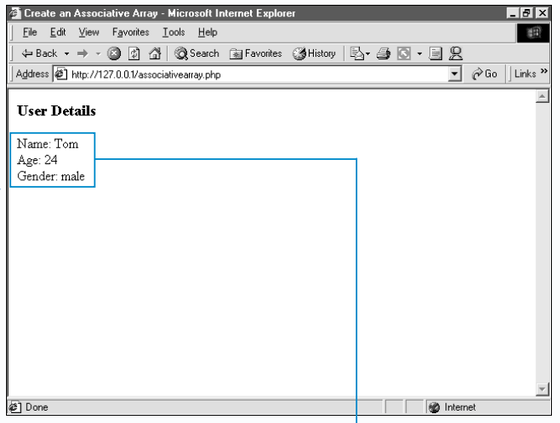
4 To access an element in the associative array, type the name of the array followed by **[]**.

5 Between the brackets, type the key of the element you want to access.



6 Type the code that uses the element in the associative array.

7 Repeat steps 4 to 6 for each element you want to access.



8 Display the PHP page in a Web browser.

- The Web browser displays the result of accessing elements in an associative array.

CREATE A MULTIDIMENSIONAL ARRAY

A multidimensional array is useful for organizing data in a grid or a table format. For example, you may use a multidimensional array to store data about different users. You create a multidimensional array by adding one or more arrays as elements of another array.

When creating a multidimensional array, you should make the array symmetrical by ensuring that each array in the multidimensional array has the same number of elements. You can use simple arrays, associative arrays or both types as elements of a multidimensional array.

To access an element in a multidimensional array, you need to specify the name of the array followed by a set of brackets for each dimension of the array. For example, accessing an element in a two-dimensional array will require two sets of brackets.

You then specify the indexes of the element you want to access in the multidimensional array within the brackets. The index of the array you want to access is specified in the first set of brackets. The index of the element within the array is specified in the second set of brackets, and so on. Keep in mind that PHP begins numbering arrays and elements at zero (0). For example, to access the third element in the second array of a two-dimensional array, you would type $values[1][2].

Accessing an element of a multidimensional array within a string, such as print "Name: $values[2][0]", may not give the correct result. To avoid problems, you should use braces { } when accessing an element of a multidimensional array within a string, such as print "Name: {$values[2][0]}". You may want to access the element outside the string and use the concatenation operator (.) to combine the strings instead, such as print "Name: " . $values[2][0].

Apply It

You can neatly display the elements of a multidimensional array in a table created using HTML. To access all the elements in the multidimensional array, you may use nested `foreach` loops. The first `foreach` loop will iterate through each array in the multidimensional array, while the other `foreach` loop will iterate through the elements in each array.

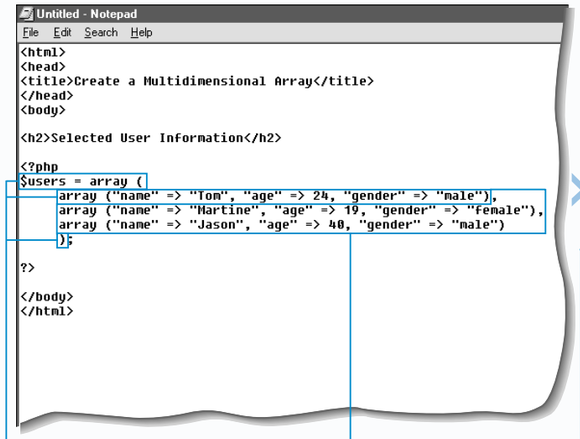
TYPE THIS:

```
$users = array (
      array ("name" => "Tom", "age" => 24, "gender" => "male"),
      array ("name" => "Martine", "age" => 19, "gender" => "female"),
      array ("name" => "Jason", "age" => 40, "gender" => "male")
      );
print "<table border=\"1\">\n";
print "<tr>\n";
print "<th>#</th>\n";
print "<th>Name</th>\n";
print "<th>Age</th>\n";
print "<th>Gender</th>\n";
print "</tr>\n";
foreach ($users as $number => $user)
{
      print "<tr><td>$number</td>";
      foreach ($user as $value)
      {
            print "<td>$value</td>";
      }
      print "</tr>";
}
print "</table>";
```

RESULT:

#	Name	Age	Gender
0	Tom	24	male
1	Martine	19	female
2	Jason	40	male

CREATE A MULTIDIMENSIONAL ARRAY

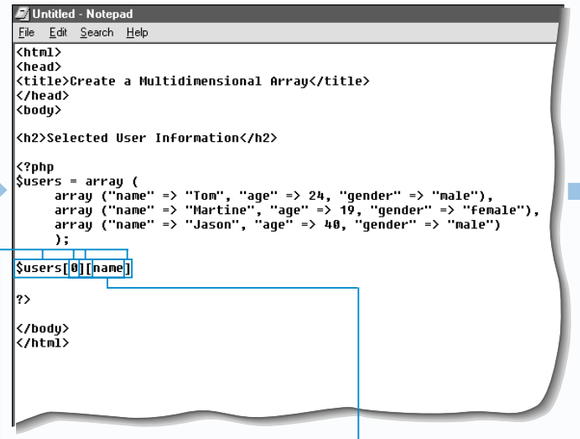


1 Type a name for the multidimensional array followed by **= array ()**.

2 To assign an array as an element of a multidimensional array, type **array** followed by the values or key-value pairs for the array enclosed in parentheses.

3 Repeat step 2 for each array you want to assign to the multidimensional array. Separate each array with a comma.

Note: In this example, a two-dimensional array is created.

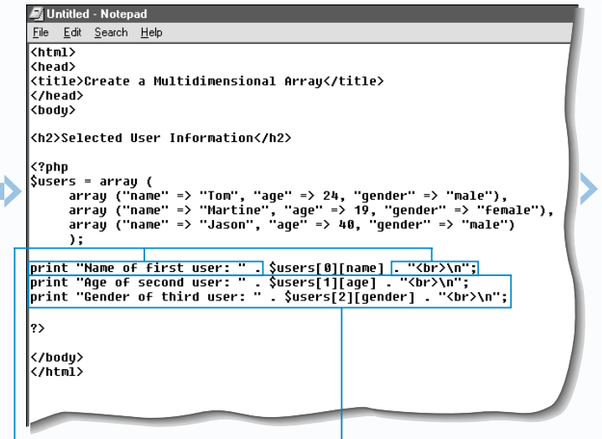


4 To access an element in the multidimensional array, type the name of the array followed by a set of brackets for each dimension in the array.

5 Between the first set of brackets, type the index of the array you want to access.

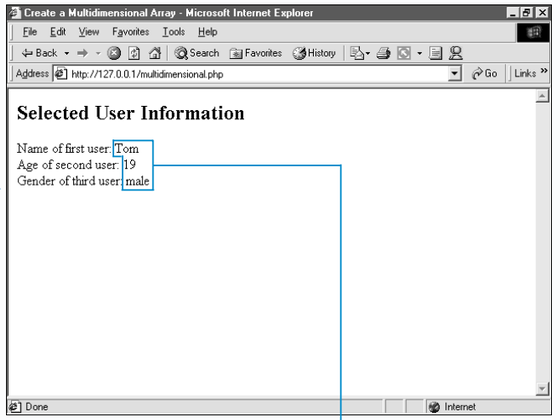
6 Between the second set of brackets, type the index or key of the element you want to access.

■ If necessary, specify additional indexes or keys within brackets.



7 Type the code that uses the element in the multidimensional array.

8 Repeat steps 4 to 7 for each element you want to access.



9 Display the PHP page in a Web browser.

■ The Web browser displays the results of accessing elements in a multidimensional array.

MOVE THROUGH ARRAY ELEMENTS

You may access the elements in an array by navigating through the array using an internal array pointer, which is used to indicate which element in the array is currently selected. This is useful when accessing array elements that have non-sequential indexes or keys. When an array is created, the pointer is set to the first element of the array. You can then move the pointer forward or back to an element you want to access.

The internal array pointer is moved using the `next`, `prev`, `end` and `reset` functions. The `next` function is used to advance the pointer to the next element in an array. The `prev` function sets the pointer back to the previous element. The `end` function sets the pointer to the last element in an array, while the `reset` function sets the pointer back to the first element in the array.

The `next`, `prev`, `end` and `reset` functions take the array you want to move through as an argument and return the value of the element to which the pointer is set.

Once you have set the pointer to the element you want to work with, you can use the `key` and `current` functions to access the element. These functions also take the name of the array as an argument. The `key` function returns the key, or index, of the current element, while the `current` function returns the value of the current element.

If the internal array pointer is advanced beyond the last element in the array or is moved back beyond the first element, the position of the pointer will be lost. You will not be able to recover the pointer using the `next` or `prev` functions. To restore the pointer, you must use the `reset` or `end` function.

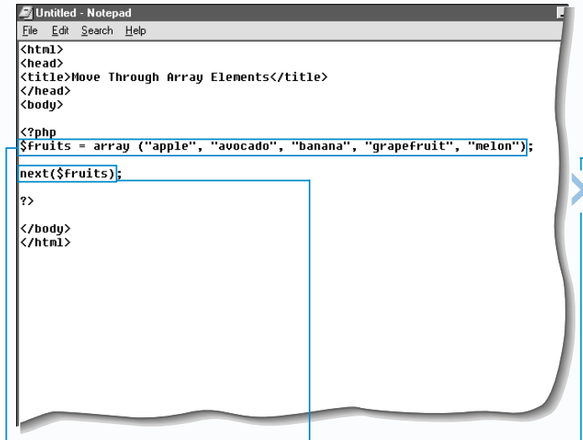
Apply It

You can create a function that advances the array pointer a specified number of times. The array and the number of advancements are passed to the function as arguments, with the array being passed by reference. In the function, a loop is used with the `next` function to advance the pointer. You can also create a function that uses the `prev` function to set the pointer back a specified number of times.

Example:

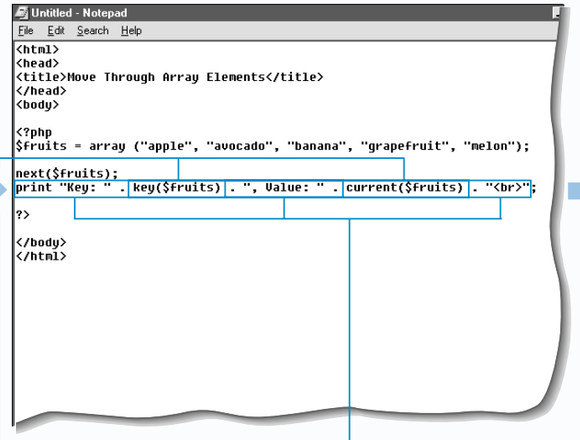
```
function AdvancePointer(&$array, $number)
{
      for ($counter = 0; $counter < $number; $counter++)
      {
            next($array);
      }
}
function MoveBackPointer(&$array, $number)
{
      for ($counter = 0; $counter < $number; $counter++)
      {
            prev($array);
      }
}
$fruits = array ("apple", "avocado", "banana", "grapefruit", "melon");
AdvancePointer($fruits, 3);
print "Key: " . key($fruits) . ", Value: " . current($fruits) . "<br>";
MoveBackPointer($fruits, 2);
print "Key: " . key($fruits) . ", Value: " . current($fruits) . "<br>";
```

MOVE THROUGH ARRAY ELEMENTS



1 Type the code that creates an array.

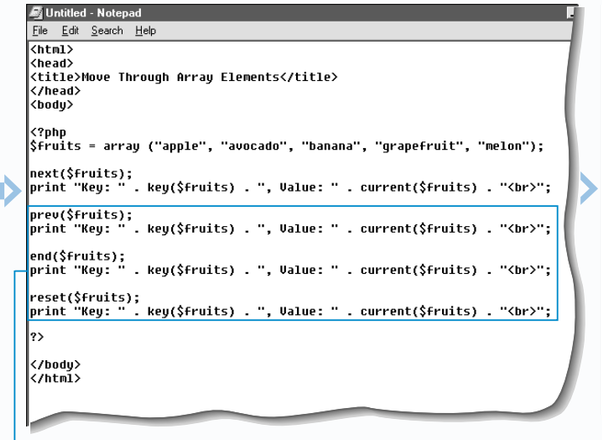
2 To move through the elements in the array, type the name of the function you want to use (**next**, **prev**, **end** or **reset**) followed by the name of the array enclosed in parentheses.



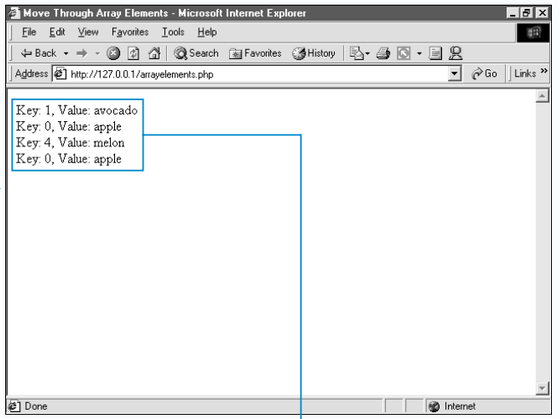
3 To access the index of the current element, type **key** followed by the name of the array enclosed in parentheses.

4 To access the value of the current element, type **current** followed by the name of the array enclosed in parentheses.

5 Type the code that uses the index and value of the current element.



6 Repeat steps 2 to 5 for each element you want to move to and access in the array.



7 Display the PHP page in a Web browser.

■ The Web browser displays the result of moving through the array elements.

ADD AND REMOVE ELEMENTS IN AN ARRAY

PHP provides several functions that allow you to modify an array by adding and removing elements in the array.

To add an element to the beginning of an array, use the `array_unshift` function. Use the `array_push` function to add an element to the end of an array. When using the `array_unshift` and `array_push` functions, you must specify the name of the array you want to modify followed by the value for the element you want to add. In addition to individual elements, the `array_unshift` and `array_push` functions also accept lists of values. This allows you to add multiple elements to the beginning or end of an array. To add more than one element, you type the value for each element, separating each value with a comma. The `array_unshift` and `array_push` functions will return the number of elements in the modified array.

To remove an element from the beginning of an array, use the `array_shift` function. The `array_pop` function allows you to remove an element from the end of an array. You can remove only one element from an array at a time using the `array_shift` or `array_pop` function. These functions will return the value of the removed element.

When you use the `array_unshift` or `array_shift` function to add or remove an element from the beginning of an array, the other elements will shift to accommodate the new element. The keys of numerically indexed elements in the array will be re-indexed starting from zero (0). Using the `array_push` or `array_pop` functions to add or remove an element from the end of an array will not affect the indexing of the other elements.

Extra

When working with arrays in PHP, it is important to remember that the position of an element in an array does not necessarily correspond to the index number of the element. The positions of array elements are determined by the order in which the elements are assigned to the array, especially in cases where the indexes of the elements are set explicitly. For example, using the `array_pop` function will remove the element that was set last in the array, not the element with the highest index number.

Example:

```
$fruits = array();
$fruits[3] = "apple";
$fruits[2] = "grape";
$fruits[1] = "melon";
$fruits[0] = "orange";
array_pop($fruits);
foreach($fruits as $value)
{
     print "$value ";
}
```

Result:

apple grape melon

You can use the assignment operator (=) to add elements to the end of an array instead of using the `array_push` function. This is useful when you want to add only a single element to the end of an array.

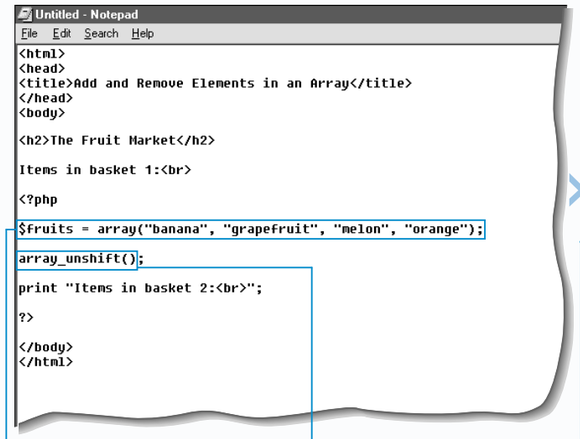
Example:

```
$fruits = array("apple", "grape", "melon");
$fruits[] = "orange";
foreach($fruits as $value)
{
     print "$value ";
}
```

Result:

apple grape melon orange

ADD AND REMOVE ELEMENTS IN AN ARRAY

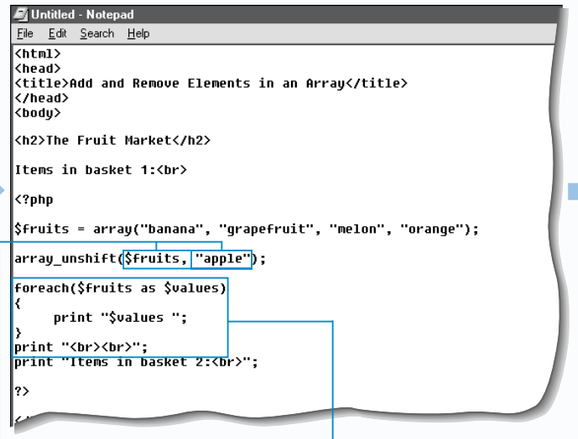


ADD AN ELEMENT

1 Type the code that creates an array.

2 To add an element to the beginning of the array, type **array_unshift()**.

■ To add an element to the end of the array, type **array_push()**.

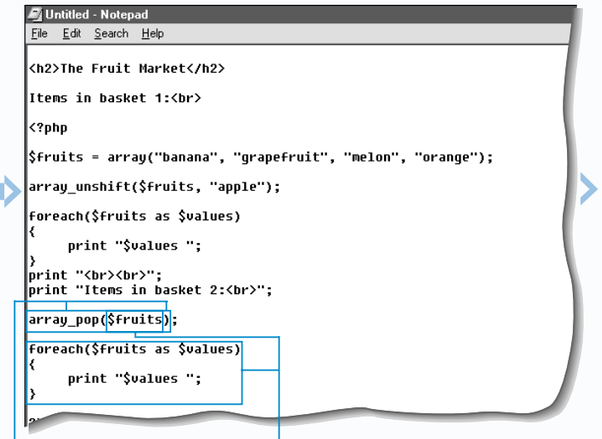


3 Between the parentheses, type the name of the array to which you want to add an element followed by a comma.

4 Type the value for the element you want to add to the array.

■ String values must be enclosed in quotation marks.

5 Type the code that uses the array.



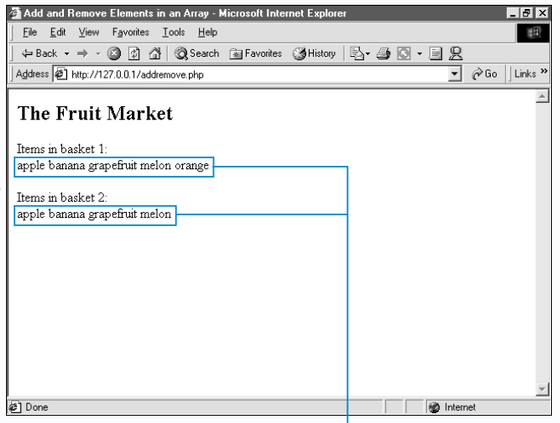
REMOVE AN ELEMENT

6 To remove the last element in the array, type **array_pop()**.

■ To remove the first element in the array, type **array_shift()**.

7 Between the parentheses, type the name of the array from which you want to remove an element.

8 Type the code that uses the array.



9 Display the PHP page in a Web browser.

■ The Web browser displays the results of adding and removing elements in an array.

REPLACE ELEMENTS IN AN ARRAY

The `array_splice` function allows you to replace existing elements in an array with new elements.

To use the `array_splice` function, you must specify the name of the array you want to change and the offset number of the first element you want to replace. The offset number corresponds to the position of the element in the array, but is not necessarily the index number of the element. The offset number will be the same as the index number of the element only if the array is indexed in consecutive order starting from zero (0).

You should also specify a length parameter to indicate the number of elements you want to replace in the array. You can replace one or more elements. If you do not specify a length parameter, PHP will replace the element represented by the offset number, followed by each element to the end of the array.

You can then indicate the new elements you want to assign to the array. The `array_splice` function allows you to replace elements in an array with an array of elements or a single element. If new elements are not specified, the elements selected for replacement in the array will simply be removed.

The `array_splice` function can also be used to add new elements to the end of an array. To add elements to the end of an array, specify an offset number that is greater than the highest index number in the array and use a length parameter of zero (0).

When you use the `array_splice` function to replace or add elements in an array, the keys of numerically indexed elements in the array will be re-indexed starting from zero (0).

When the `array_splice` function is executed, an array containing the replaced array elements is returned.

Extra

You can use a negative offset number to specify the position of the first element you want to replace. When a negative number is used, the `array_splice` function finds the position of the elements from the end of the array, starting at 1.

TYPE THIS:

```
$fruits = array("apple", "avocado", "banana", "grapefruit", "melon");
array_splice($fruits, -2, 2, "peach");
foreach($fruits as $value)
{
     print "$value ";
}
```

RESULT:

apple avocado banana peach

If you want to work with only some of the elements in an array, you can extract the elements you want without modifying the array. To extract elements from an array, use the `array_slice` function instead of the `array_splice` function.

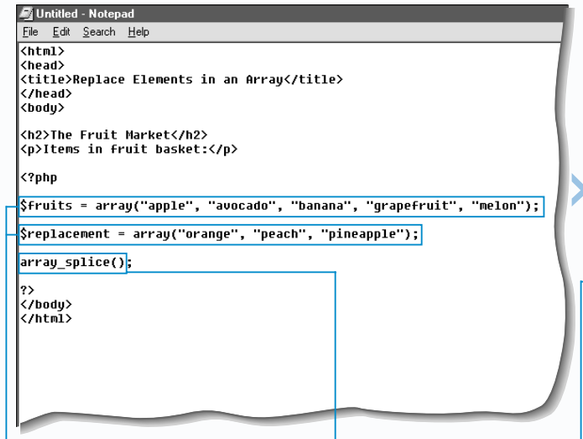
TYPE THIS:

```
$fruits = array("apple", "avocado", "banana", "grapefruit", "melon");
$extracted = array_slice($fruits, 1, 3);
foreach($extracted as $value)
{
     print "$value"."s ";
}
```

RESULT:

avocados bananas grapefruits

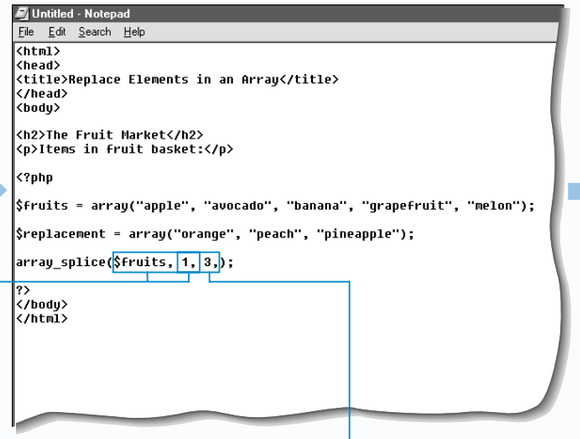
REPLACE ELEMENTS IN AN ARRAY



1 Type the code that creates an array.

2 To replace elements in the array with more than one new element, type the code that creates an array that contains the new elements.

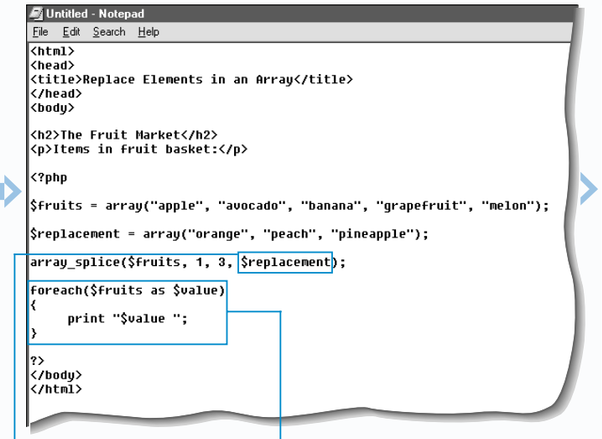
3 To replace elements in an array, type **array_splice()**.



4 Between the parentheses, type the name of the array that contains elements you want to replace followed by a comma.

5 Type the offset number of the first element you want to replace followed by a comma.

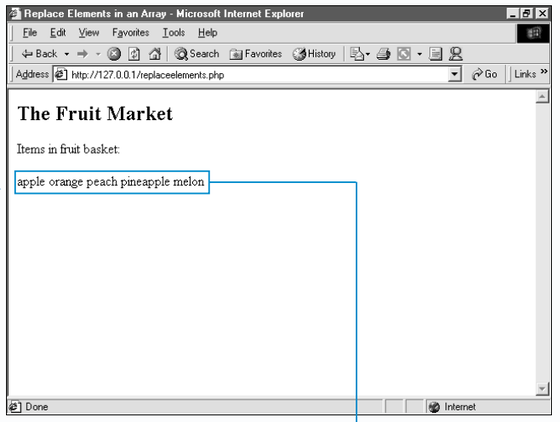
6 Type the number of elements you want to replace followed by a comma.



7 To specify the elements you want to assign to the array, type the name of the array that contains the new elements.

■ To replace the array elements with just one element, you can type the value for the new element. A string value must be enclosed in quotation marks.

8 Type the code that uses the array.



9 Display the PHP page in a Web browser.

■ The Web browser displays the result of replacing elements in an array.

SORT A SIMPLE ARRAY

The `sort` function allows you to sort a simple array and place the elements in order from lowest to highest. For example, you could sort an array of numbers into ascending order or sort an array of strings into alphabetical order.

The `sort` function takes the name of the array you want to sort as its argument. When you sort an array using the `sort` function, the keys for numerically indexed elements will be re-indexed starting from zero (0). This makes the `sort` function unsuitable for sorting an associative array. For information about sorting associative arrays, see page 96.

PHP allows you to use an optional second argument with the `sort` function to specify how the data values should be treated. The `SORT_REGULAR` value specifies that PHP should evaluate the array elements based on their ASCII values. By default, PHP will use the `SORT_REGULAR` value when sorting an array, unless another sort type is specified. To specify that you want the array elements to be evaluated as numbers, you can use the `SORT_NUMERIC` value. If you specify the `SORT_STRING` value, the elements will be evaluated as strings.

Specifying a sort type argument is useful when you want to force PHP to evaluate array elements in a specific way. For example, PHP would normally sort an array containing the numbers 1, 2 and 10 as 1 10 2. To ensure that the numbers are sorted in correct ascending order, you should use the `SORT_NUMERIC` value.

The `sort` function will return a value of true or false depending on whether the sort was executed successfully.

To sort an array in reverse order, from the highest to the lowest value, you use the `rsort` function. The `rsort` function has the same properties and takes the same arguments as the `sort` function.

Extra

When PHP sorts strings, values beginning with uppercase characters are grouped together first, followed by values beginning with lowercase characters. Using the same case for all the array elements will ensure that they are sorted in the correct order. You can use the `strtolower` function to convert the values in an array to all lowercase characters before performing a sort.

TYPE THIS:

```
$fruits = array("Orange", "apple", "Banana",
 "mango", "Grape");
foreach($fruits as $key => $value)
{
      $fruits[$key] = strtolower($value);
}
sort($fruits);
foreach($fruits as $value)
{
      print "$value ";
}
```

RESULT:

apple banana grape mango orange

You can arrange the elements in an array in random order. To arrange array elements in random order, you use the `shuffle` function. Before using the `shuffle` function, you must call the `srand` function to initialize PHP's random number generator.

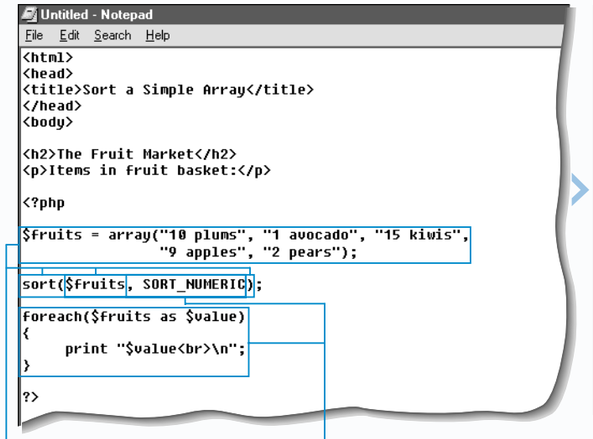
TYPE THIS:

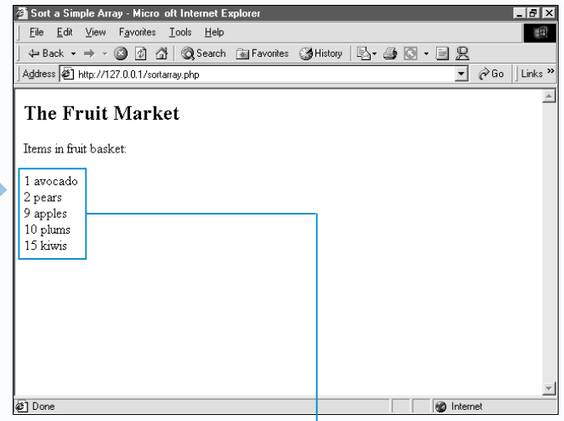
```
srand((double) microtime() * 1000000);
$numbers = array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
shuffle($numbers);
foreach($numbers as $value)
{
      print "$value ";
}
```

RESULT:

1 2 3 7 0 4 5 6 8 10 9

SORT A SIMPLE ARRAY





1 Type the code that creates an array.

2 To sort the array, type **sort()**.

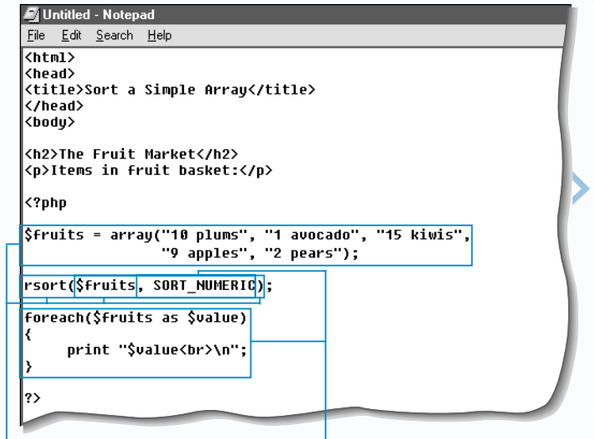
3 Between the parentheses, type the name of the array you want to sort.

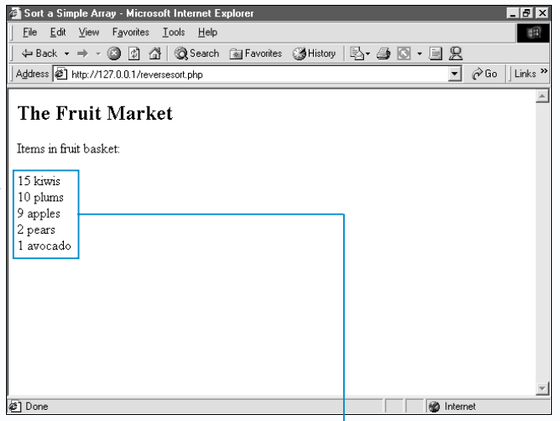
4 To specify the way you want PHP to evaluate the elements, type a comma followed by a sort value.

5 Type the code that uses the array.

6 Display the PHP page in a Web browser.

The Web browser displays the results of sorting the values in an array.





SORT AN ARRAY IN REVERSE ORDER

1 Type the code that creates an array.

2 To sort the array in reverse order, type **rsort()**.

3 Between the parentheses, type the name of the array you want to sort in reverse order.

4 To specify the way you want PHP to evaluate the elements, type a comma followed by a sort value.

5 Type the code that uses the array.

6 Display the PHP page in a Web browser.

The Web browser displays the results of sorting the values in an array in reverse order.

SORT AN ASSOCIATIVE ARRAY

The `asort` function is used to sort an associative array so that the relationship between the key and value of the array elements are preserved during the sorting process. When the `asort` function is used, the values of an associative array are sorted from lowest to highest. Using the `asort` function to sort the values in an associative array ensures that the array elements will not be re-indexed starting from 0, which would cause the original keys to be lost.

You may also sort an associative array based on the keys of the array elements rather than the values using the `ksort` function. When the `ksort` function is used, the keys will be sorted from lowest to highest while maintaining the relationship between the key and value of the array elements.

The `asort` and `ksort` functions take the associative array you want to sort as an argument and will return a value of true or false depending on whether the sort was executed successfully.

PHP allows you to use an optional second argument with the `asort` and `ksort` functions to specify how the data values should be treated. The `SORT_REGULAR` value specifies that PHP should evaluate the array elements based on their ASCII values. By default, PHP will use the `SORT_REGULAR` value when sorting an array, unless another sort type is specified. To specify that you want the array elements to be evaluated as numbers, you can use the `SORT_NUMERIC` value. If you specify the `SORT_STRING` value, the elements will be evaluated as strings.

Extra

The `natsort` function sorts alphanumeric values using a method called natural ordering, which is based on how a person would naturally order items. The `natsort` function is based on Martin Pool's Natural Order String Comparison algorithm. You can find more information about natural ordering at the www.linuxcare.com.au/projects/natsort Web site. The `natsort` function can be used to sort both simple and associative arrays.

TYPE THIS:

```
$images = array ("img100.gif", "img12.gif",
                 "img10.gif", "img2.gif",
                 "img1.gif", "img20.gif");
natsort($images);
foreach ($images as $value)
{
    print "$value  ";
}
```

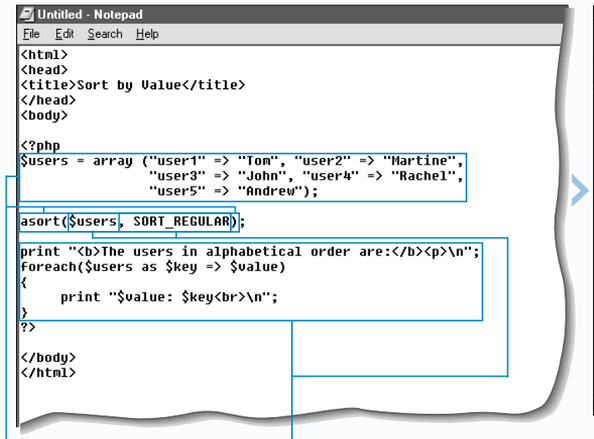
RESULT:

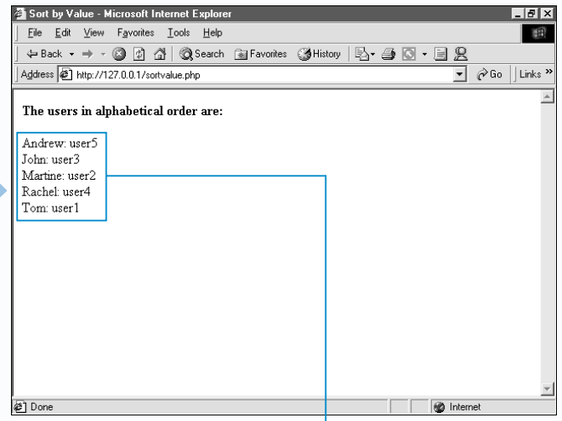
img1.gif img2.gif img10.gif img12.gif img20.gif img100.gif

The `natcasesort` function also uses the natural ordering method to sort values, but it sorts values in a case-insensitive manner. Similar to the other functions used to sort associative arrays, the `natcasesort` function maintains the relationship between the key and value of the array elements.

The `arsort` function is used to sort the values of an associative array from highest to lowest. To sort an associative array based on keys from highest to lowest, the `krsort` function is used. The `arsort` and `krsort` functions take the associative array you want to sort as an argument. You can also specify the sort type you wish to use to evaluate the values or keys.

SORT AN ASSOCIATIVE ARRAY

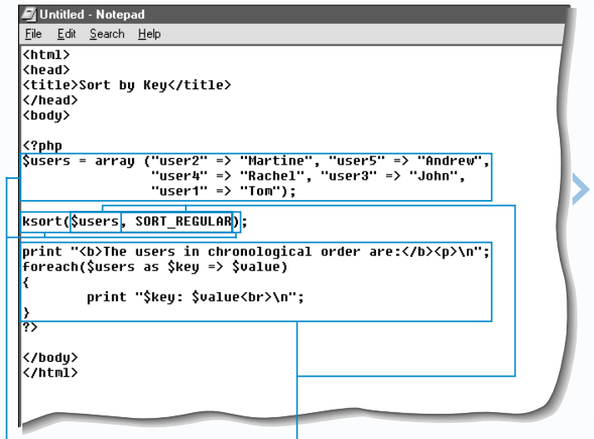


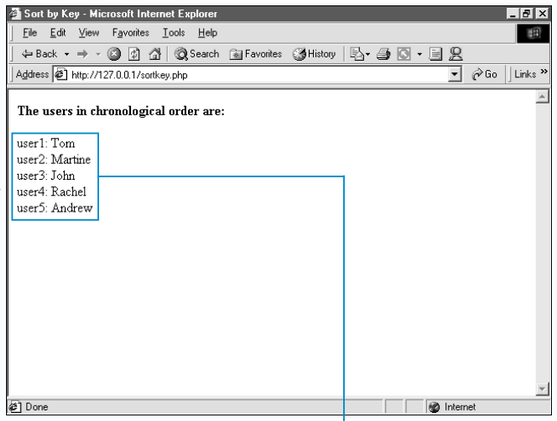


SORT BY VALUE

1. Type the code that creates an associative array.
2. To sort the array according to the value of each element, type **asort()**.
3. Between the parentheses, type the name of the array you want to sort.
4. To specify the way you want PHP to evaluate the values, type a comma followed by a sort value.
5. Type the code that uses the array.
6. Display the PHP page in a Web browser.

■ The Web browser displays the result of sorting an associative array by the value of each element.





SORT BY KEY

1. Type the code that creates an associative array.
2. To sort the array according to the value of each element, type **ksort()**.
3. Between the parentheses, type the name of the array you want to sort.
4. To specify the way you want PHP to evaluate the keys, type a comma followed by a sort value.
5. Type the code that uses the array.
6. Display the PHP page in a Web browser.

■ The Web browser displays the result of sorting an associative array by the key of each element.

SORT AN ARRAY USING COMPARISON FUNCTIONS

You can create a function that compares values in an array to determine in which order the values should be sorted. To avoid unnecessary work, you should check whether there is an existing sorting function that suits your needs before defining your own function.

A comparison function takes two arguments representing two adjacent values in an array. The function compares the two values and returns a value of -1, 1 or 0. A value of -1 indicates that the first value will appear first in the sorted array. A value of 1 indicates that the second value will appear first in the sorted array. If the function returns a value of 0, the two values are treated equally in the sort order.

The code in the body of a comparison function typically contains a conditional structure, which is used to compare the two values and determine the value to be returned.

Once the comparison function has been defined, you can use the `usort` function to sort the values in an array. When using the `usort` function, the array you want to sort is passed as the first argument. The name of the comparison function is passed as the second argument and must be enclosed in quotation marks. The array will then be sorted based on the order determined by the comparison function. When an array is sorted using the `usort` function, the array elements are re-indexed starting from zero (0).

To maintain the association between the key and value of the array elements, such as when sorting an associative array, you should use the `uasort` or `uksort` function instead of the `usort` function. The `uasort` function sorts an array based on the value of the array elements. The `uksort` function sorts an array based on the key of the array elements. The `uasort` and `uksort` functions take the same arguments as the `usort` function.

Apply It

The code in the comparison function may contain more than one conditional structure, depending on how the two values are being compared. For example, in a comparison function used to compare two numbers, the code may initially compare the two values so that the odd number is first. If the two values are both odd numbers or both even numbers, the code may then compare the two values so that the smaller number is first.

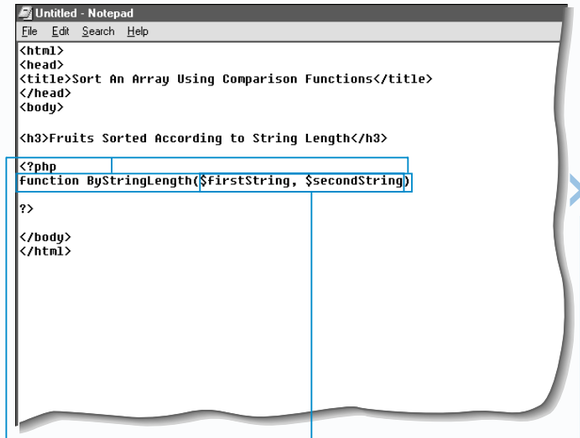
TYPE THIS:

```
function OddEven($firstValue, $secondValue)
{
      $firstRem = $firstValue % 2;
      $secondRem = $secondValue % 2;
      if (($firstRem == 1) && ($secondRem == 0)) return -1;
      else if (($firstRem == 0) && ($secondRem == 1)) return 1;
      else
      {
            if ($firstValue < $secondValue) return -1;
            else if ($firstValue > $secondValue) return 1;
            else return 0;
      }
}
$numbers = array(18, 5, 9, 3, 69, 101, 58, 93, 14, 80, 28);
usort($numbers, "OddEven");
foreach ($numbers as $value)
{
      print "$value ";
}
```

RESULT:

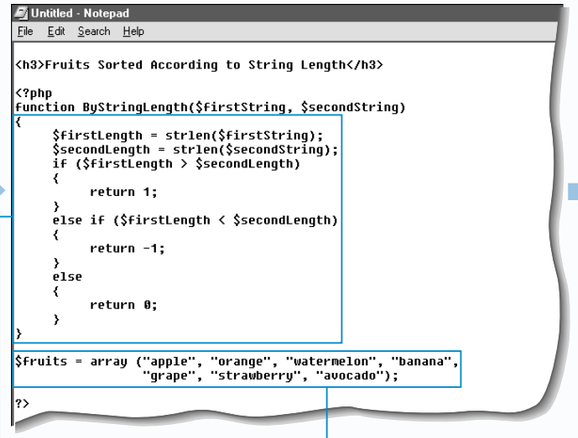
```
3 5 9 69 93 101 14 18 28 58 80
```

SORT AN ARRAY USING COMPARISON FUNCTIONS



1 To create a comparison function, type **function** followed by a name for the comparison function and ().

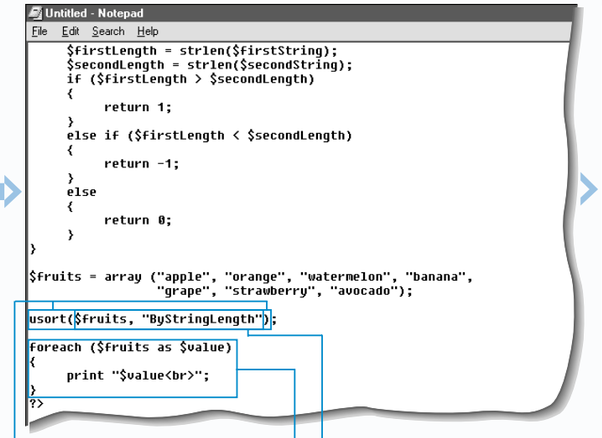
2 Between the parentheses, type the name of two variables that represent two adjacent values in an array, separated by a comma.



3 In the body of the function, type the code that compares the two values and returns a value of -1, 1 or 0, depending on the result of the comparison.

■ In this example, the comparison function compares the length of two string values and places the string with the shorter length first.

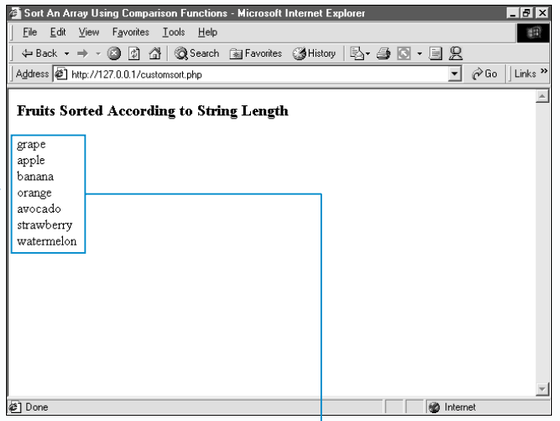
4 Type the code that creates an array.



5 Type the name of the function you want to use to sort the array (**usort**, **uasort** or **uksort**) followed by ().

6 Between the parentheses, type the name of the array followed by a comma. Then type name of the comparison function enclosed in quotation marks.

7 Type the code that uses the sorted array.



8 Display the PHP page in a Web browser.

■ The Web browser displays the result of using a user-defined comparison function to sort an array.

GET INFORMATION ABOUT AN ARRAY

There are several functions available that you can use to find information about an array.

The `array_count_values` function is used to count the number of times a value occurs in an array. This function returns an array in which each key represents a unique value in the initial array. The value associated with a key indicates the number of times the value occurred in the initial array. This is useful for calculating subtotals in an array.

The `array_sum` function is used to calculate the sum of all the values in an array and is typically used on an array of numeric values. The `array_sum` function returns the sum as an integer or a floating-point number.

You can use the `array_keys` function to obtain all the keys in an array. The `array_keys` function returns an array containing all the keys.

The `array_reverse` function is used to reverse the order of the elements in an array. When using the `array_reverse` function, the elements in the resulting array are re-indexed starting from 0.

The `array_flip` function is used to switch the keys and values in an array. The `array_flip` function returns an array in which the keys are the values from the initial array and the values are the corresponding keys from the initial array.

These functions all take the array you want to work with as an argument. The result of each function is typically assigned to a variable that can then be displayed on a Web browser or further processed.

The `in_array` function is also used to find information about an array. This function allows you to find a specific value in an array. The `in_array` function takes two arguments—the value you are looking for and the array to be searched. The `in_array` function will return a value of true if the specified value is found.

Extra

You can use the `array_rand` function to retrieve random entries from an array. Before using the `array_rand` function, you need to use the `srand` function to initialize PHP's random number generator. When using the `array_rand` function, you must specify the array that contains the elements you want to select from. You may also specify the number of elements you want to choose.

TYPE THIS:

```
srand((double)microtime() * 10000000);
$numbers = array ("one", "two", "three", "four",
                  "five", "six", "seven", "eight", "nine");
$pickThree = array_rand($numbers, 3);
foreach ($pickThree as $value)
{
      print "$numbers[$value] ";
}
```

RESULT:

four nine seven

The `array_unique` function can be used to remove duplicate values from an array by returning an array containing only the unique values. This function is useful for removing redundant values that have been inadvertently added to an array.

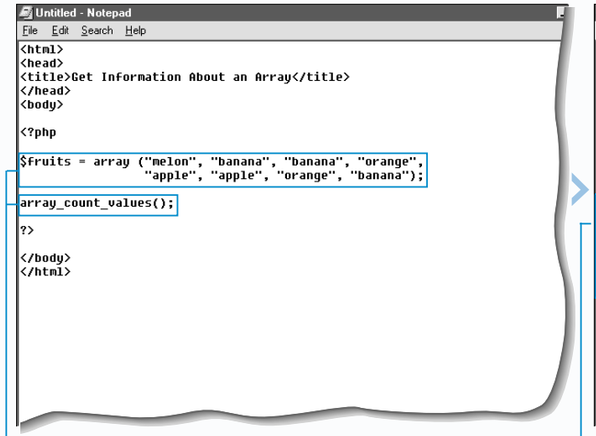
TYPE THIS:

```
$fruits = array ("apple", "banana", "apple",
                 "banana", "orange");
$unique = array_unique($fruits);
foreach ($unique as $value)
{
      print "$value ";
}
```

RESULT:

apple banana orange

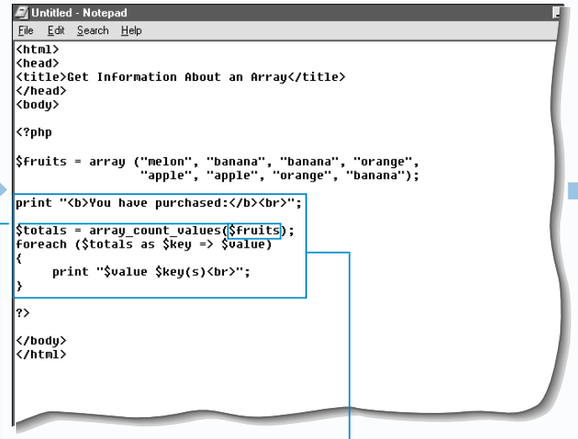
GET INFORMATION ABOUT AN ARRAY



1 Type the code that creates an array.

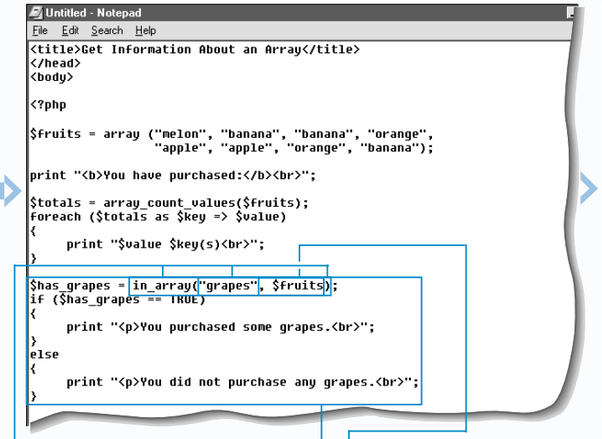
2 Type the function you want to use to find information about the array followed by ().

■ You can use the `array_count_values`, `array_sum`, `array_keys`, `array_reverse` or `array_flip` function.



3 Between the parentheses, type the name of the array you want to use.

4 Type the code that uses the result of the function.



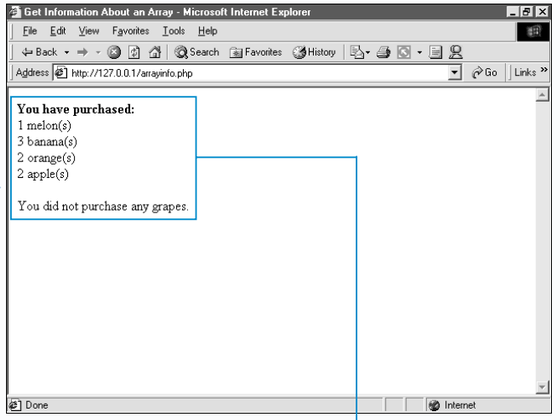
USING THE `in_array` FUNCTION

5 To determine if a value is found in an array, type **in_array()**.

6 Between the parentheses, type the value you want to find, enclosed in quotation marks.

7 Type a comma followed by the name of the array you want to search.

8 Type the code that uses the result of `in_array` function.



9 Display the PHP page in a Web browser.

■ The Web browser displays the results of obtaining information about an array.

USING THE LIST STATEMENT

PHP allows you to use the `list` statement to assign the values in an array to a list of variables. Using the `list` statement can make your arrays easier to work with and is especially useful when working with functions that return an array. Instead of referencing an array element by its numeric key, you can assign the element to a variable with a meaningful name. When you want to use the element, you can use the variable name to access the element. This can also make your code easier to read and understand.

The `list` statement takes as its arguments the names of the variables to which you want to assign the values in an array, separated by commas. You can assign all, some or just one of the values in an array to a variable. For example, if you want to retrieve only the third element in an array, you do not need to assign all the elements to variables. Instead, you can leave the spaces for the first two elements empty and specify a variable name for only the third element, such as list(, , $day).

When using the `list` statement, you should use only arrays with numerically indexed elements starting from zero (0). The order of the variables you specify in the `list` statement should correspond to the index numbers of the values, not the positions of the values in the array.

The `list` statement cannot access values in an array with an associative index. You also cannot use the `list` statement to assign a value that is not part of an array to a variable.

Extra

The `each` function is commonly used with the `list` statement to retrieve the key and value of an element in an array. The `each` function returns an array that has an index of 0 for the key and an index of 1 for the value of the element at which the internal array pointer is currently positioned. The pointer is then advanced to the next element. Code that creates a loop, such as a `while` statement, is required to access the key and value of each element in an array.

TYPE THIS:

```
$user = array("Name" => "Frank", "Age" => "33", "Gender" => "male");
while (list($key, $value) = each($user))
{
     print "$key: $value<br>\n";
}
```

RESULT:

Name: Frank
Age: 33
Gender: male

You can use the `next`, `prev`, `end` and `reset` functions to move through the elements in an array. This allows you to move the internal array pointer to a specific element you want to access in an array. You can then use the each function with the `list` statement to assign the value of the element to a variable.

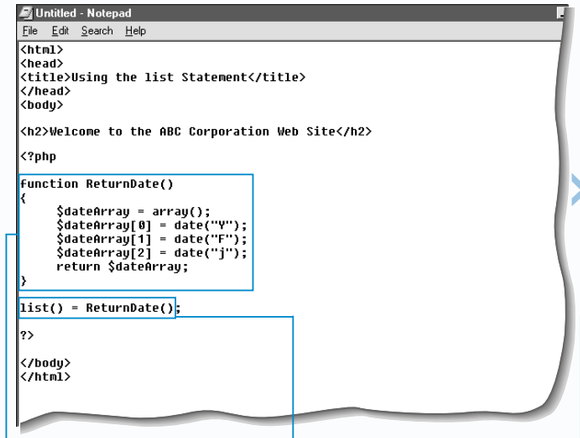
TYPE THIS:

```
$user = array("Name" => "Frank", "Age" => "33", "Gender" => "male");
next($user);
next($user);
list(, $gender) = each($user);
print "$gender";
```

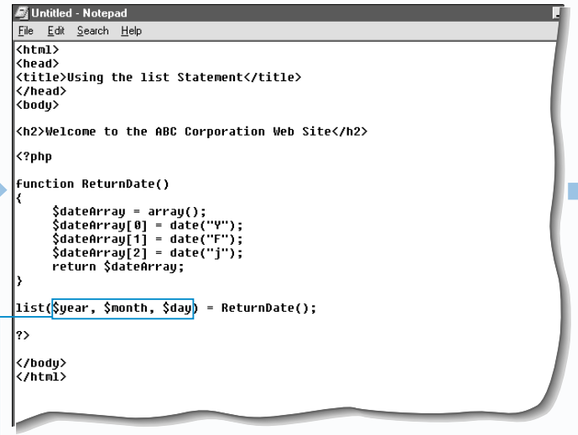
RESULT:

male

USING THE LIST STATEMENT



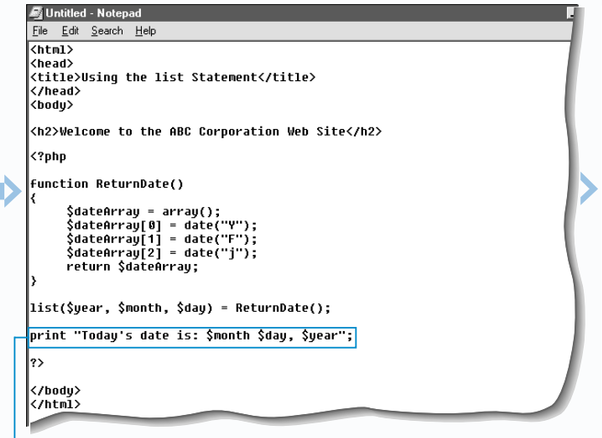
1 Type the code that creates a function that returns an array.



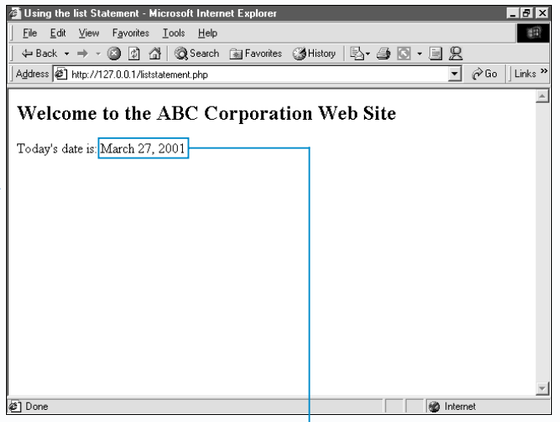
2 To assign the values in the array to a list of variables, type **list()**. Then type **=** followed by the name of the function that returns the array.

3 Between the parentheses for the `list` function, type the name of each variable to which you want to assign a value in the array. Separate each variable name with a comma.

■ If you do not want to assign a value to a variable, leave a blank space for the value.



4 Type the code that uses the variables.



5 Display the PHP page in a Web browser.

■ The Web browser displays the result of using the `list` statement to assign array elements to variables.