Sorting arrays

PHP provides two handy functions to convert between variable arrays and strings because they are so frequently used together. The **implode()** function converts an array to a string and inserts a specified separator between each value. Typically, this might be used to create a list of comma-separated values like this:

```
$csv_list = implode( ' , ' , $array ) ;
```

Conversely, the **explode()** function converts a string to an array by specifying a separator around which to break up the string. This might be used to create an array from a comma-separated list:

```
$array = explode( ' , ' , $csv_list ) ;
```

PHP also provides three useful functions to sort array elements into ascending alphanumeric order (a-z 1-9):

- **sort()** function – sorts by value discarding the original key
- **asort()** function – sorts by value retaining the original key
- **ksort()** function – sorts by key

Existing array values are sorted with a simple statement like this:

```
$makers = array( 'Ford' , 'Chevrolet' , 'Dodge' ) ;
sort( $makers ) ;
```

Array elements can also be sorted into descending alphanumeric order (9-1 z-a) with three similar functions:

- **rsort()** function – sorts by value discarding the original key
- **arsort()** function – sorts by value retaining the original key
- **krsort()** function – sorts by key

It is important to recognize that specified key names will be discarded by the **sort()** and **rsort()** functions so these should only be used where the key/value relationship is not significant – otherwise use the **asort()** and **arsort()** functions to retain the keys.

You can test if a variable is an array using the **is_array()** function. For example **is_array($var) ;**