```
$aref2 = $aref1;
```

You get two references to the same array. If you modify `$aref1->[23]` and then look at `$aref2->[23]` you'll see the change.

To copy the array, use

```
$aref2 = [@{$aref1}];
```

This uses `[...]` notation to create a new anonymous array, and `$aref2` is assigned a reference to the new array. The new array is initialized with the contents of the array referred to by `$aref1`.

Similarly, to copy an anonymous hash, you can use

```
$href2 = {%{$href1}};
```

- To see if a variable contains a reference, use the `ref` function. It returns true if its argument is a reference. Actually it's a little better than that: It returns `HASH` for hash references and `ARRAY` for array references.
- If you try to use a reference like a string, you get strings like

```
ARRAY(0x80f5dec)   or   HASH(0x826afc0)
```

If you ever see a string that looks like this, you'll know you printed out a reference by mistake.

A side effect of this representation is that you can use `eq` to see if two references refer to the same thing. (But you should usually use == instead because it's much faster.)

- You can use a string as if it were a reference. If you use the string `"foo"` as an array reference, it's taken to be a reference to the array `@foo`. This is called a *soft reference* or *symbolic reference*. The declaration `use strict 'refs'` disables this feature, which can cause all sorts of trouble if you use it by accident.

You might prefer to go on to *perllol* instead of *perlref*; it discusses lists of lists and multidimensional arrays in detail. After that, you should move on to *perldsc*; it's a Data Structure Cookbook that shows recipes for using and printing out arrays of hashes, hashes of arrays, and other kinds of data.

3.7 Summary

Everyone needs compound data structures, and in Perl the way you get them is with references. There are four important rules for managing references: Two for making references and two for using them. Once you know these rules you can do most of the important things you need to do with references.

3.8 Credits

Author: Mark Jason Dominus, Plover Systems (`mjd-perl-ref+@plover.com`)

This article originally appeared in *The Perl Journal* (http://www.tpj.com/) volume 3, #2. Reprinted with permission.

The original title was *Understand References Today*.

3.8.1 Distribution Conditions