Perl hash in scalar and list context

hash

Prev

Next

We have seen how to create a hash from an array, by assigning the array to the hash. Can we do it in the other way around? What will happen if we assign a hash to an array? What will happen if we assign a hash to a scalar?

In this example we have a hash with 3 pairs. First we assign it to an array, then to a scalar variable and then we print out the results. We use Data::Dumper to serialize the hash and the array, but we print the scalar directly.

examples/hash_in_context.pl

```
use strict;
use warnings;
use 5.010;
use Data::Dumper qw(Dumper);

my %color_of = (
    apple  => 'red',
    grape  => 'purple',
    banana => 'yellow',
);

my @pairs = %color_of;
my $var = %color_of;

say Dumper \%color_of;
say Dumper \@pairs;
say $var;
```

If we run the script this is what we get:

```
$ perl examples/hash_in_context.pl
$VAR1 = {
          'grape' => 'purple',
          'apple' => 'red',
          'banana' => 'yellow'
        };

$VAR1 = [
          'grape',
          'purple',
          'apple',
          'red',
          'banana',
          'yellow'
        ];

3/8
```

If we run it again we get this:

```
$ perl examples/hash_in_context.pl
$VAR1 = {
          'banana' => 'yellow',
          'apple' => 'red',
          'grape' => 'purple'
        };

$VAR1 = [
          'banana',
          'yellow',
          'apple',
          'red',
          'grape',
          'purple'
        ];

2/8
```

In both results the first `$VAR1` represents the original hash, the second `$VAR1` represents the array. As you can see the content of the hash is the same in both cases, but the order is different. In the case of that hash this change in the order is only in the representation, when we converted the hash into something we can print. After all, inside the hash there is no order.

The order of the values is different between the two arrays as well. This, on the other hand is the actual difference in the result. When we assign the hash to an array, we basically put the hash in `LIST context`. When a hash is in LIST context Perl converts a hash into a list of alternating values. Each key-value pair in the original hash will become two values in the newly created list. For every pair the key will come first and the value will come after. In which order the pairs will be included in the resulting list is not defined.

So in the above case we can be sure that 'yellow' will come right after 'banana', and that 'red' will come right after 'apple', but we don't know if the order will be `'banana', 'yellow', 'apple', 'red'` or `'apple', 'red', 'banana', 'yellow'`. Moreover, every time we run the code the order might be different. If we have more pairs, the number of possibilities is just much larger.

So we know the hash will be flattened, and we know each pair will be in the expected order, but we don't know in what order the pairs will return.

Hash in scalar context

The last value in both printouts is the value of the **hash in scalar context**. This is actually some internal number representing the internal layout of the hash. It is not very useful except for the fact that this value will be 0 only if the hash itself is empty. So we can write code like this:

```
if (%color_of) {
    ...
}
```

and we can be sure that the block will be executed only if the hash was **not** empty.