23.1.3 Do I always/never have to quote my strings or use semicolons and commas?

Normally, a bareword doesn't need to be quoted, but in most cases probably should be (and must be under `use strict`). But a hash key consisting of a simple word (that isn't the name of a defined subroutine) and the left-hand operand to the => operator both count as though they were quoted:

```
This                    is like this
------------            ---------------
$foo{line}              $foo{"line"}
bar => stuff            "bar" => stuff
```

The final semicolon in a block is optional, as is the final comma in a list. Good style (see *perlstyle*) says to put them in except for one-liners:

```
if ($whoops) { exit 1 }
@nums = (1, 2, 3);

if ($whoops) {
    exit 1;
}
@lines = (
    "There Beren came from mountains cold",
    "And lost he wandered under leaves",
);
```

23.1.4 How do I skip some return values?

One way is to treat the return values as a list and index into it:

```
    $dir = (getpwnam($user))[7];
```

Another way is to use undef as an element on the left-hand-side:

```
($dev, $ino, undef, undef, $uid, $gid) = stat($file);
```

You can also use a list slice to select only the elements that you need:

```
    ($dev, $ino, $uid, $gid) = ( stat($file) )[0,1,4,5];
```

23.1.5 How do I temporarily block warnings?

If you are running Perl 5.6.0 or better, the `use warnings` pragma allows fine control of what warning are produced. See *perllexwarn* for more details.

```
{
    no warnings;          # temporarily turn off warnings
    $a = $b + $c;         # I know these might be undef
}
```

If you have an older version of Perl, the `$^W` variable (documented in *perlvar*) controls runtime warnings for a block:

```
{
    local $^W = 0;        # temporarily turn off warnings
    $a = $b + $c;         # I know these might be undef
}
```

Note that like all the punctuation variables, you cannot currently use my() on `$^W`, only local().