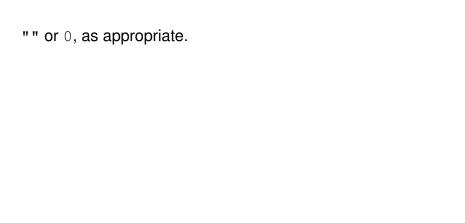
A newline character.

- \$ scalar@ array
- % hash
- & subroutine
- * typeglob

Double quotes perform variable interpolation and backslash

interpretation. Single quotes do not.

Back ticks capture the output from executing a command.



As the expected type, depending on context.

interpolative context

list context

```
@threeprimes = (2, 3, 5);
($a, $b, $c) = @threeprimes;
```

Either involves a scalar so use \$ not @.

```
$1st[n] = new_el
print $1st[n]
```

```
@birthmonths = (
    "John" => "February",
    "Mary" => "March",
);
```

Arrows are just a nicer way of writing more arrows.

```
$hash{"key"}
$hash{"key"} = val
```

As with arrays, notice the use of \$ when dealing with individual elements.

Nouns can be singular (scalars) or plural (arrays and hashes). Verbs can be procedures or functions.

perl -e 'some perl'
perl file.pl

The -w option prints warnings.

A data type that can represent files, devices, sockets, and pipes.

STDOUT and STDERR are provided by default.

Use open, whose simplest form is:

open(HANDLENAME, "filename");

Readonly (default): "<filename"
Write (clobber): ">filename"
Write (append): ">>filename"

```
$str = <FILEHANDLE>
$str = <STDIN>
```

print FILEHANDLE 'str'

These two are the same:

print STDOUT 'str'
print 'str'

chop removes the last character of the string passed to it, and returns it.

chomp removes endl from the string passed to it, and returns the *number* of characters removed.

Use a period (.) for string concatenation.

sum.

Because of weak typing addition of scalars created as strings, but that can be interpreted as numbers, would result in a