28.1.19 Conditional Operator

Ternary "?:" is the conditional operator, just as in C. It works much like an if-then-else. If the argument before the ? is true, the argument before the : is returned, otherwise the argument after the : is returned. For example:

```
printf "I have %d dog%s.\n", $n,
        ($n == 1) ? '' : "s";
```

Scalar or list context propagates downward into the 2nd or 3rd argument, whichever is selected.

```
$a = $ok ? $b : $c;  # get a scalar
@a = $ok ? @b : @c;  # get an array
$a = $ok ? @b : @c;  # oops, that's just a count!
```

The operator may be assigned to if both the 2nd and 3rd arguments are legal lvalues (meaning that you can assign to them):

```
($a_or_b ? $a : $b) = $c;
```

Because this operator produces an assignable result, using assignments without parentheses will get you in trouble. For example, this:

```
$a % 2 ? $a += 10 : $a += 2
```

Really means this:

```
(($a % 2) ? ($a += 10) : $a) += 2
```

Rather than this:

```
($a % 2) ? ($a += 10) : ($a += 2)
```

That should probably be written more simply as:

```
$a += ($a % 2) ? 10 : 2;
```

28.1.20 Assignment Operators

"=" is the ordinary assignment operator.

Assignment operators work as in C. That is,

```
$a += 2;
```

is equivalent to

```
$a = $a + 2;
```

although without duplicating any side effects that dereferencing the lvalue might trigger, such as from tie(). Other assignment operators work similarly. The following are recognized:

```
**=    +=    *=    &=    <<=    &&=
       -=    /=    |=    >>=    ||=
       .=    %=    ^=
             x=
```