## POSIX Character Classes

Perl supports the POSIX notation for character classes. This uses names enclosed by `[:` and `:]` within the enclosing brackets. PCRE also supports this notation. For example,

```
[01[:alpha:]%]
```

matches "0", "1", any alphabetic character, or "%".

The supported class names, all of which must appear between `[:` and `:]` inside a character class specification, are

| | |
|---|---|
| `alnum` | Letters and digits. |
| `alpha` | Letters. |
| `ascii` | Character codes 0 - 127. |
| `blank` | Space or tab only. |
| `cntrl` | Control characters. |
| `digit` | Decimal digits (same as `\d`). |
| `graph` | Printing characters, excluding space. |
| | [:graph:] matches all characters with the Unicode L, M, N, P, S, or Cf properties with a few arcane exceptions. |
| `lower` | Lower case letters. |
| `print` | Printing characters, including space. |
| | [:print:] matches the same characters as [:graph:] plus space characters that are not controls, that is, characters with the Unicode Zs property. |
| `punct` | Printing characters, excluding letters and digits. |
| | [:punct:] matches all characters that have the Unicode P (punctuation) property, plus those characters whose code points are less than 128 that have the S (Symbol) property. |
| `space` | White space (not quite the same as `\s`). |
| `upper` | Upper case letters. |
| `word` | "Word" characters (same as `\w`). |
| `xdigit` | Hexadecimal digits. |

The "space" characters are horizontal tab (`HT-9`), linefeed (`LF-10`), vertical tab (`VT-11`), formfeed (`FF-12`), carriage-return (`CR-13`), and space (`32`).

The class name `word` is a Perl extension, and `blank` is a GNU extension from Perl 5.8. Another Perl extension is negation that is indicated by a `^` character after the colon. For example,

```
[12[:^digit:]]
```

matches "1", "2", or any nondigit. PCRE (and Perl) also recognize the POSIX syntax `[.ch.]` and `[=ch=]` where "ch" is a "collating element", but these are not supported, and an error is given if they are encountered.

## Alternation

Vertical bar characters are used to separate alternative patterns. For example, the pattern

```
gilbert|sullivan
```

matches either "gilbert" or "sullivan". Any number of alternative patterns may be specified, and an empty alternative is permitted (matching the empty string). The matching process tries each alternative in turn, from left to right, and the first one that succeeds is used. If the alternatives are within a subpattern (defined