

NAME

feature - Perl pragma to enable new features

SYNOPSIS

```
use feature qw(switch say);
given ($foo) {
    when (1)    { say "\$foo == 1" }
    when ([2,3]) { say "\$foo == 2 || \$foo == 3" }
    when (/^a[bc]d$/) { say "\$foo eq 'abd' || \$foo eq 'acd'" }
    when ($_ > 100) { say "\$foo > 100" }
    default    { say "None of the above" }
}

use feature ':5.10'; # loads all features available in perl 5.10
```

DESCRIPTION

It is usually impossible to add new syntax to Perl without breaking some existing programs. This pragma provides a way to minimize that risk. New syntactic constructs, or new semantic meanings to older constructs, can be enabled by `use feature 'foo'`, and will be parsed only when the appropriate feature pragma is in scope.

Lexical effect

Like other pragmas (`use strict`, for example), features have a lexical effect. `use feature qw(foo)` will only make the feature "foo" available from that point to the end of the enclosing block.

```
{
    use feature 'say';
    say "say is available here";
}
print "But not here.\n";
```

no feature

Features can also be turned off by using `no feature "foo"`. This too has lexical effect.

```
use feature 'say';
say "say is available here";
{
    no feature 'say';
    print "But not here.\n";
}
say "Yet it is here.";
```

`no feature` with no features specified will turn off all features.

The 'switch' feature

`use feature 'switch'` tells the compiler to enable the Perl 6 `given/when` construct.

See *"Switch statements" in perlsyn* for details.

The 'say' feature

`use feature 'say'` tells the compiler to enable the Perl 6 `say` function.

See *"say" in perlfunc* for details.

the 'state' feature

`use feature 'state'` tells the compiler to enable `state` variables.

See "*Persistent Private Variables*" in *perlsub* for details.

the 'unicode_strings' feature

`use feature 'unicode_strings'` tells the compiler to treat all strings outside of `use locale` and `use bytes` as Unicode. It is available starting with Perl 5.11.3.

See "*The 'Unicode Bug'*" in *perlunicode* for details.

FEATURE BUNDLES

It's possible to load a whole slew of features in one go, using a *feature bundle*. The name of a feature bundle is prefixed with a colon, to distinguish it from an actual feature. At present, the only feature bundle is `use feature ":5.10"` which is equivalent to `use feature qw(switch say state)`.

Specifying sub-versions such as the 0 in 5.10.0 in feature bundles has no effect: feature bundles are guaranteed to be the same for all sub-versions.

IMPLICIT LOADING

There are two ways to load the `feature` pragma implicitly :

- By using the `-E` switch on the command-line instead of `-e`. It enables all available features in the main compilation unit (that is, the one-liner.)
- By requiring explicitly a minimal Perl version number for your program, with the `use VERSION` construct, and when the version is higher than or equal to 5.10.0. That is,

```
use 5.10.0;
```

will do an implicit

```
use feature ':5.10';
```

and so on. Note how the trailing sub-version is automatically stripped from the version.

But to avoid portability warnings (see "*use*" in *perlfunc*), you may prefer:

```
use 5.010;
```

with the same effect.