NAME

Module::Load - runtime require of both modules and files

SYNOPSIS

```
use Module::Load;

my $module = 'Data::Dumper';

load Data::Dumper; # loads that module, but not import any functions
# -> cannot use 'Dumper' function

load 'Data::Dumper'; # ditto
load $module # tritto

autoload Data::Dumper; # loads that module and imports the default functions
# -> can use 'Dumper' function

my $script = 'some/script.pl'
load $script;
load 'some/script.pl'; # use quotes because of punctuations

load thing; # try 'thing' first, then 'thing.pm'

load CGI, ':all'; # like 'use CGI qw[:standard]'
```

DESCRIPTION

Module::Load eliminates the need to know whether you are trying to require either a file or a module.

If you consult perldoc(1) -f requirey ou will see that `require` will behave differently when given a bareword or a string.

In the case of a string, `require` assumes you are wanting to load a file. But in the case of a bareword, it assumes you mean a module.

This gives nasty overhead when you are trying to dynamically require modules at runtime, since you will need to change the module notation (`Acme::Comment`) to a file notation fitting the particular platform you are on.

Module::Load eliminates the need for this overhead and will just DWYM.

Difference between `load` and `autoload`

Module::Load imports the two functions - `load` and `autoload`

`autoload` imports the default functions automatically, but `load` do not import any functions.

`autoload` is usable under `BEGIN{};`.

Both the functions can import the functions that are specified.

Following codes are same.

```
load File::Spec::Functions, qw/splitpath/;

autoload File::Spec::Functions, qw/splitpath/;
```

FUNCTIONS

load

Loads a specified module.

See ‘‘Rules’’ for detailed loading rule.

autoload
Loads a specified module and imports the default functions.

Except importing the functions, 'autoload' is same as 'load'.

load_remote
Loads a specified module to the specified package.

```
use Module::Load 'load_remote';

my $pkg = 'Other::Package';

load_remote $pkg, 'Data::Dumper'; # load a module to 'Other::Package'
# but do not import 'Dumper' function
```

A module for loading must be quoted.

Except specifing the package and quoting module name, 'load_remote' is same as 'load'.

autoload_remote
Loads a specified module and imports the default functions to the specified package.

```
use Module::Load 'autoload_remote';

my $pkg = 'Other::Package';

autoload_remote $pkg, 'Data::Dumper'; # load a module to 'Other::Package'
# and imports 'Dumper' function
```

A module for loading must be quoted.

Except specifing the package and quoting module name, 'autoload_remote' is same as 'load_remote'.

Rules

All functions have the following rules to decide what it thinks you want:

- If the argument has any characters in it other than those matching `\w`, `:` or `'`, it must be a file
- If the argument matches only `[\w:']`, it must be a module
- If the argument matches only `\w`, it could either be a module or a file. We will try to find `file.pm` first in `@INC` and if that fails, we will try to find `file` in `@INC`. If both fail, we die with the respective error messages.

IMPORTS THE FUNCTIONS

'load' and 'autoload' are imported by default, but 'load_remote' and 'autoload_remote' are not imported.

To use 'load_remote' or 'autoload_remote', specify at 'use'.

''load'',''autoload'',''load_remote'',''autoload_remote''
Imports the selected functions.

```
# imports 'load' and 'autoload' (default)
use Module::Load;

# imports 'autoload' only
use Module::Load 'autoload';

# imports 'autoload' and 'autoload_remote', but don't import 'load';
use Module::Load qw/autoload autoload_remote/;
```

'all'
Imports all the functions.

```
use Module::Load 'all'; # imports load, autoload, load_remote, autoload_remote
```

'','none',undef
Not import any functions (`load` and `autoload` are not imported).

```
use Module::Load '';

use Module::Load 'none';

use Module::Load undef;
```

Caveats

Because of a bug in perl (#19213), at least in version 5.6.1, we have to hardcode the path separator for a require on Win32 to be `/`, like on Unix rather than the Win32 `\`. Otherwise perl will not read its own `%INC` accurately double load files if they are required again, or in the worst case, core dump.

`Module::Load` cannot do implicit imports, only explicit imports. (in other words, you always have to specify explicitly what you wish to import from a module, even if the functions are in that modules' `@EXPORT`)

ACKNOWLEDGEMENTS

Thanks to Jonas B. Nielsen for making explicit imports work.

BUG REPORTS

Please report bugs or other issues to <bug-module-load@rt.cpan.org<gt>.

AUTHOR

This module by Jos Boumans <kane@cpan.org>.

COPYRIGHT