NAME

Clone - recursively copy Perl datatypes

SYNOPSIS

```
use Clone 'clone';

my $data = {
  set => [ 1 .. 50 ],
  foo => {
     answer => 42,
     object => SomeObject->new,
  },
};

my $cloned_data = clone($data);

$cloned_data->{foo}{answer} = 1;
print $cloned_data->{foo}{answer};  # '1'
print $data->{foo}{answer};         # '42'
```

You can also add it to your class:

```
package Foo;
use parent 'Clone';
sub new { bless {}, shift }

package main;

my $obj = Foo->new;
my $copy = $obj->clone;
```

DESCRIPTION

This module provides a "clone()" method which makes recursive copies of nested hash, array, scalar and reference types, including tied variables and objects.

"clone()" takes a scalar argument and duplicates it. To duplicate lists, arrays or hashes, pass them in by reference, e.g.

```
my $copy = clone (\@array);
```

```
    # or

    my %copy = %{ clone (\%hash) };
```

SEE ALSO

Storable's "dclone()" is a flexible solution for cloning variables, albeit slower for average-sized data structures. Simple and naive benchmarks show that Clone is faster for data structures with 3 or fewer levels, while "dclone()" can be faster for structures 4 or more levels deep.

COPYRIGHT

AUTHOR

Ray Finch "<rdf@cpan.org>"

Breno G. de Oliveira "<garu@cpan.org>", Nicolas Rochelemagne "<atoomic@cpan.org>" and Florian Ragwitz "<rafl@debian.org>" perform routine maintenance releases since 2012.