NAME

Object::ID – A unique identifier for any object

SYNOPSIS

```
package My::Object;

# Imports the object_id method
use Object::ID;
```

DESCRIPTION

This is a unique identifier for any object, regardless of its type, structure or contents. Its features are:

```
* Works on ANY object of any type
* Does not modify the object in any way
* Does not change with the object's contents
* Is O(1) to calculate (ie. doesn't matter how big the object is)
* The id is unique for the life of the process
* The id is always a true value
```

USAGE

Object::ID is a role, rather than inheriting its methods they are imported into your class. To make your class use Object::ID, simply `use Object::ID` in your class.

```
package My::Class;

use Object::ID;
```

Then write your class however you want.

METHODS

The following methods are made available to your class.

object_id

```
my $id = $object->object_id;
```

Returns an identifier unique to the `$object`.

The identifier is not related to the content of the object. It is only unique for the life of the process. There is no guarantee as to the format of the identifier from version to version.

For example:

```
my $obj = My::Class->new;
my $copy = $obj;

# This is true, $obj and $copy refer to the same object
$obj->object_id eq $copy->object_id;

my $obj2 = My::Class->new;

# This is false, $obj and $obj2 are different objects.
$obj->object_id eq $obj2->object_id;

use Clone;
my $clone = clone($obj);

# This is false, even though they contain the same data.
$obj->object_id eq $clone->object_id;
```

object_uuid

```
my $uuid = $object->object_uuid
```

Like `$object->object_id` but returns a UUID unique to the `$object`.

Only works if Data::UUID is installed.

See Data::UUID for more details about UUID.

FAQ

Why not just use the object's reference?

References are not unique over the life of a process. Perl will reuse references of destroyed objects, as demonstrated by this code snippet:

```
{
    package Foo;

    sub new {
        my $class = shift;
        my $string = shift;
        return bless {}, $class;
    }
}

for(1..3) {
    my $obj = Foo->new;
    print "Object's reference is $obj\n";
}
```

This will print, for example, `Object's reference is Foo=HASH(0x803704)` three times.

How much memory does it use?

Very little.

Object::ID stores the ID and address of each object you've asked the ID of. Once the object has been destroyed it no longer stores it. In other words, you only pay for what you use. When you're done with it, you don't pay for it any more.

LICENSE

THANKS

Thank you to Vincent Pit for coming up with the implementation.