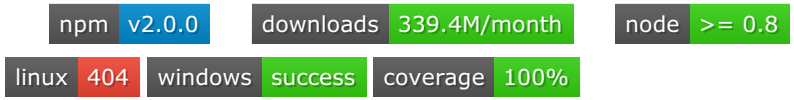
depd



Deprecate all the things

With great modules comes great responsibility; mark things deprecated!

Install

This module is installed directly using `npm`:

```
$ npm install depd
```

This module can also be bundled with systems like Browserify or webpack, though by default this module will alter it's API to no longer display or track deprecations.

API

```
var deprecate = require('depd')('my-
        module')
```

This library allows you to display deprecation messages to your users. This library goes above and beyond with deprecation warnings by introspection of the call stack (but only the bits that it is interested in).

Instead of just warning on the first invocation of a deprecated function and never again, this module will warn on the first invocation of a deprecated function per unique call site, making it ideal to alert users of all deprecated uses across the code base, rather than just whatever happens to execute first.

The deprecation warnings from this module also include the file and line information for the call into the module that the deprecated function was in.

NOTE this library has a similar interface to the `debug` module, and this module uses the calling file to get the boundary for the call stacks, so you should always create a new `deprecate` object in each file and not within some central file.

depd(namespace)

Create a new deprecate function that uses the given namespace name in the messages and will display the call site prior to the stack entering the file this function was called from. It is highly suggested you use the name of your module as the namespace.

License

MIT

```
    // calls with non-string first
        argument are deprecated
    deprecate('weirdfunction non-string
        first arg')
  }
}
```

Deprecating property access

This will display a deprecated message about “oldprop” being deprecated from “my-module” on STDERR when accessed. A deprecation will be displayed when setting the value and when getting the value.

```
var deprecate = require('depd')('my-
        cool-module')

exports.oldprop = 'something'

// message automatically derives from
        property name
deprecate.property(exports, 'oldprop')

// explicit message
deprecate.property(exports, 'oldprop',
        'oldprop >= 0.10')
```


deprecate(message)

Call this function from deprecated code to display a deprecation message. This message will appear once per unique caller site. Caller site is the first call site in the stack in a different file from the caller of this function.

If the message is omitted, a message is generated for you based on the site of the `deprecate()` call and will display the name of the function called, similar to the name displayed in a stack trace.

deprecate.function(fn, message)

Call this function to wrap a given function in a deprecation message on any call to the function. An optional message can be supplied to provide a custom message.

deprecate.property(obj, prop, message)

Call this function to wrap a given property on object in a deprecation message on any accessing or setting of the property. An optional message can be supplied to provide a custom message.

The method must be called on the object where the property belongs (not inherited from the prototype).

If the property is a data descriptor, it will be converted to an accessor descriptor in order to display the deprecation message.

process.on('deprecation', fn)

This module will allow easy capturing of deprecation errors by emitting the errors as the type "deprecation" on the global `process`. If there are no listeners for this type, the errors are written to STDERR as normal, but if there are any listeners, nothing will be written to STDERR and instead only emitted. From there, you can write the errors in a different format or to a logging source.

The error represents the deprecation and is emitted only once with the same rules as writing to STDERR. The error has the following properties:

- `message` - This is the message given by the library
- `name` - This is always `'DeprecationError'`
- `namespace` - This is the namespace the deprecation came from
- `stack` - This is the stack of the call to the deprecated thing

Example `error.stack` output:

```
DeprecationError: my-cool-module
deprecated oldfunction
    at Object.<anonymous> ([eval]-
wrapper:6:22)
    at Module._compile
(module.js:456:26)
    at evalScript (node.js:532:25)
```


Conditionally deprecating a function call

This will display a deprecated message about "weirdfunction" being deprecated from "my-module" on STDERR when called with less than 2 arguments.

```
var deprecate = require('depd')('my-
        cool-module')

exports.weirdfunction = function () {
  if (arguments.length < 2) {
    // calls with 0 or 1 args are
        deprecated
    deprecate('weirdfunction args < 2')
  }
}
```

When calling `deprecate` as a function, the warning is counted per call site within your own module, so you can display different deprecations depending on different situations and the users will still get all the warnings:

```
var deprecate = require('depd')('my-
        cool-module')

exports.weirdfunction = function () {
  if (arguments.length < 2) {
    // calls with 0 or 1 args are
        deprecated
    deprecate('weirdfunction args < 2')
  } else if (typeof arguments[0] !==
        'string') {
```

Examples

Deprecating all calls to a function

This will display a deprecated message about “oldfunction” being deprecated from “my-module” on STDERR.

```
var deprecate = require('depd')('my-
        cool-module')

// message automatically derived from
        function name
// Object.oldfunction
exports.oldfunction =
        deprecate.function(function
        oldfunction () {
  // all calls to function are
        deprecated
})

// specific message
exports.oldfunction =
        deprecate.function(function () {
  // all calls to function are
        deprecated
}, 'oldfunction')
```



```
    at startup (node.js:80:7)
    at node.js:902:3
```

process.env.NO_DEPRECATION

As a user of modules that are deprecated, the environment variable `NO_DEPRECATION` is provided as a quick solution to silencing deprecation warnings from being output. The format of this is similar to that of `DEBUG`:

```
$ NO_DEPRECATION=my-module,othermod node
        app.js
```

This will suppress deprecations from being output for “my-module” and “othermod”. The value is a list of comma-separated namespaces. To suppress every warning across all namespaces, use the value * for a namespace.

Providing the argument `--no-deprecation` to the `node` executable will suppress all deprecations (only available in Node.js 0.8 or higher).

NOTE This will not suppress the deperecations given to any “deprecation” event listeners, just the output to STDERR.

process.env.TRACE_DEPRECATION

As a user of modules that are deprecated, the environment variable `TRACE_DEPRECATION` is provided as a solution to getting more detailed location information in deprecation

warnings by including the entire stack trace. The format of this is the same as `NO_DEPRECATION`:

```
$ TRACE_DEPRECATION=my-module,othermod
        node app.js
```

This will include stack traces for deprecations being output for “my-module” and “othermod”. The value is a list of comma-separated namespaces. To trace every warning across all namespaces, use the value `*` for a namespace.

Providing the argument `--trace-deprecation` to the `node` executable will trace all deprecations (only available in Node.js 0.8 or higher).

NOTE This will not trace the deperecations silenced by `NO_DEPRECATION`.

Display

When a user calls a function in your library that you mark deprecated, they will see the following written to STDERR (in the given colors, similar colors and layout to the `debug` module):

```
bright cyan    bright yellow
|              |          reset
cyan
|              |          |          |
▼              ▼          ▼          ▼
```



```
my-cool-module deprecated oldfunction
[eval]-wrapper:6:22
▲              ▲          ▲          ▲
|              |          |          |
namespace      |          |
location of mycoolmod.oldfunction() call
               |          deprecation
message
               the word "deprecated"
```

If the user redirects their STDERR to a file or somewhere that does not support colors, they see (similar layout to the `debug` module):

```
Sun, 15 Jun 2014 05:21:37 GMT my-cool-
module deprecated oldfunction at [eval]-
wrapper:6:22
▲
▲              ▲          ▲
▲
|
|              |          |
|
timestamp of message
namespace      |          |
location of mycoolmod.oldfunction() call

|          deprecation message

the word "deprecated"
```