delegates

Node method and accessor delegation utilty.

Installation

```
$ npm install delegates
```

Example

```
var delegate = require('delegates');

...

delegate(proto, 'request')
  .method('acceptsLanguages')
  .method('acceptsEncodings')
  .method('acceptsCharsets')
  .method('accepts')
  .method('is')
  .access('querystring')
  .access('idempotent')
  .access('socket')
  .access('length')
  .access('query')
  .access('search')
  .access('status')
  .access('method')
  .access('path')
  .access('body')
  .access('host')
  .access('url')
  .getter('subdomains')
  .getter('protocol')
  .getter('header')
  .getter('stale')
  .getter('fresh')
  .getter('secure')
  .getter('ips')
  .getter('ip')
```

API

Delegate(proto, prop)

Creates a delegator instance used to configure using the `prop` on the given `proto` object. (which is usually a prototype)

Delegate#method(name)

Allows the given method `name` to be accessed on the host.

Delegate#getter(name)

Creates a "getter" for the property with the given `name` on the delegated object.

Delegate#setter(name)

Creates a "setter" for the property with the given `name` on the delegated object.

Delegate#access(name)

Creates an "accessor" (ie: both getter *and* setter) for the property with the given `name` on the delegated object.

Delegate#fluent(name)

A unique type of "accessor" that works for a "fluent" API. When called as a getter, the method returns the expected value. However, if the method is called with a value, it will return itself so it can be chained. For example:

```
delegate(proto, 'request')
  .fluent('query')

// getter
var q = request.query();

// setter (chainable)
request
  .query({ a: 1 })
  .query({ b: 2 });
```

License

MIT