MODULES AND NPM

MODULES

ANATOMY OF A MODULE

A module is a file.

A module is a JavaScript file.

A module is a JavaScript file that exports an object.

Example:

```
module.exports = function getPI() {
  return Math.PI;
};
```

A **unit of logic** such as:

```
var pi = Math.PI;

function diameter(radius) {
  return2 * radius;
}

function circumference(radius) {
  return pi * diameter(radius);
}

function area(radius) {
  return pi * Math.pow(radius, 2);
}
```

CAN BE A MODULE:

...with private code

01_circle.js:

```
var pi = Math.PI;

module.exports = function(radius) {

  function diameter() {
    return 2 * radius;
  }

  function circumference() {
    return pi * diameter();
  }

  function area() {
    return pi * Math.pow(radius, 2);
  }

  return {
    area: area,
    circumference: circumference
  };
}
```

USING MODULES

FROM THE REPL

```
$ node
> _
```

```
> var Circle = require('./01_circle.js');
undefined
> var circle = Circle(10);
undefined
> circle
{ area: [Function: area],
  circumference: [Function: circumference] }
> > circle.area()
314.1592653589793
> circle.circumference()
62.83185307179586
```

You can also omit the `.js`:

```
> var Circle = require('./01_circle');
```

JAVASCRIPT CONSTRUCTORS

are functions!

Uppercased `Circle` informs the reader of the intent of the function.

REQUIRE

`require()` is a qualified import:

- Does not modify the environment (returns the module exported value)
- Resolves to a full file path, not conflicting with any other module

OTHER GLOBALS

- `__filename` - path to the current file
- `__dirname` - path to the current file's directory

USER MODULES

`require` can be called with a file path (absolute or relative to the current file):

```
require('./lib/shapes/circle.js');
require('../util.js');
```

CORE MODULES

`require` can be called with the name of a core module:

```
var http = require('http');
var fs = require('fs');
```

There are several core modules like `http`, `net`, `assert`, `dgram`, `path`, `url`, `util` and others.

MODULE CACHING

```
$ node
> var request = require('request');
> request.hello = "world";
> console.log(require('request').hello);
world
undefined
```

Modules are only loaded from disk once

NPM

COMMAND LINE NPM

a Unix-like tool

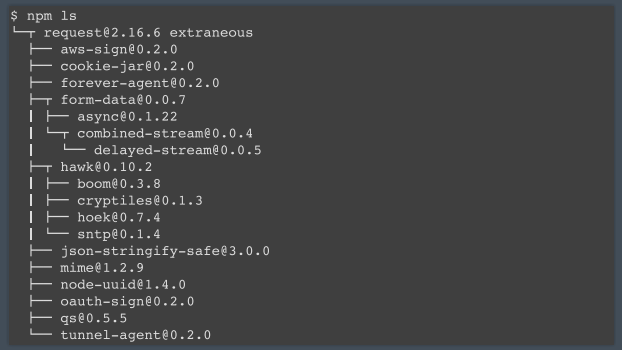
```
$ npm search request
$ npm doc request
$ npm install request
$ npm remove request
```

INSTALLING MODULES

npm is bundled with Node.js.

Install the request module:

```
$ npm install request
```


$ npm ls
└─┬ request@2.16.6 extraneous
  ├── aws-sign@0.2.0
  ├── cookie-jar@0.2.0
  ├── forever-agent@0.2.0
  ├─┬ form-data@0.0.7
  │ ├── async@0.1.22
  │ └─┬ combined-stream@0.0.4
  │   └── delayed-stream@0.0.5
  ├─┬ hawk@0.10.2
  │ ├── boom@0.3.8
  │ ├── cryptiles@0.1.3
  │ ├── hoek@0.7.4
  │ └── sntp@0.1.4
  ├── json-stringify-safe@3.0.0
  ├── mime@1.2.9
  ├── node-uuid@1.4.0
  ├── oauth-sign@0.2.0
  ├── qs@0.5.5
  └── tunnel-agent@0.2.0

NPM installs in a local `node_modules` folder:

```
$ ls node_modules
request
```

Back in the REPL:

```
$ node
> var request = require('request');
```

Module search paths for module names (not a file path)

1. core module
2. `node_modules` folder in current directory
3. `node_modules` folder in parent directories

PACKAGE.JSON

the package manifest

package.json:

```
{
  "name": "modules-src",
  "version": "0.0.1",
  "description": "Modules unit source code",
  "dependencies": {
    "request": "*",
    "async": "*"
  }
}
```

Now, you can tell NPM to resolve those dependencies locally:

```
$ npm install
```

this will ensure all the dependencies in your package.json are installed

```
$ node
> var async = require('async');
```

SUMMARY

Modules **separate** program logic into discrete, manageable units.