Node Patterns for module.exports

There are several patterns which developers can use to export functionality from a module:

- Export a function: `module.exports = function(ops) {...}`
- Export an object: `module.exports = {...}`
- Export multiple functions: `module.exports.methodA = function(ops) {...}` which is the same as `exports.methodA = function(ops) {...}`
- Export multiple objects: `module.exports.objA = {...}` which is the same as `exports.objA = {...}`

`module.exports.name = ...` or `exports.name =...` are used for multiple export points in a single file. They are equivalent to using `module.exports = {name: ...}`.

Be careful! `exports = ...` (without `module`) is *not* a valid module/export statement.

Let's take a look at how to export and use multiple functions. We begin with a monolithic program named `print-greetings.js` that has all the logic to print hello in three languages.

print-greetings.js:

```
// greeting methods
var sayHelloInEnglish = function() {
  return 'Hello'
}

var sayHelloInSwedish = function() {
  return 'Hej'
}

var sayHelloInTatar = function() {
  return 'Isänme'
}

console.log('Swedish ' +
  sayHelloInSwedish() +
  ' & English ' +
  sayHelloInEnglish() +
  ' & Tatar ' +
  sayHelloInTatar())
```

You can see that if you add 50 more languages, this file will start to become difficult to manage. If you wanted to use the `sayHello...` methods in other files, then this monolithic file wouldn't work well either. Let's modularize the program by putting the translation methods into their own module named `greetings.js`.

Exporting methods using `exports.methodA = function(ops) {...}`

We can export the greeting methods by individually defining the greeting methods on the `exports` object.

greetings.js:

```
exports.sayHelloInEnglish = function() {
  return 'Hello'
}

exports.sayHelloInSwedish = function() {
  return 'Hej'
}

exports.sayHelloInTatar = function() {
  return 'Isänme'
}
```

Exporting methods using `module.exports = {...}`

We can also export the greeting methods by setting `module.exports` equal to an object that contains the greeting methods.

greetings.js:

```
module.exports = {
  sayHelloInEnglish() {
    return 'Hello'
  }

  sayHelloInSwedish() {
    return 'Hej'
  }

  sayHelloInTatar() {
    return 'Isänme'
  }
}
```

Regardless of the export pattern you use, `module.exports` will end up being an object with three greeting methods.

Importing with require()

Next, we can edit `print-greetings.js` to import methods from `greetings.js` using `require()`. The `require()` method returns whatever was exported from the imported module. In this case, the `require()` method returns an object with three greeting methods and that object gets assigned to the `greetings` variable. The greetings methods are then accessible through the `greetings` variable.

printGreetings.js:

```
var greetings = require('./greetings.js')

console.log('Swedish ' +
  greetings.sayHelloInSwedish() +
  ' & English ' +
  greetings.sayHelloInEnglish() +
  ' & Tatar ' +
  greetings.sayHelloInTatar())
```