**CommonJS specification:**

The Node.js module system is an implementation of the **CommonJS** specification. CommonJS describes a simple syntax for JavaScript programs to require (or import) other JavaScript programs into their context. This missing feature of JavaScript greatly assists with creating modular systems by simplifying the process of separating concerns. In Node.js, all JavaScript files can be seen as individual modules. So, beyond this point, we'll use the terms: file and module interchangeably. We may have also heard the term package being used in the place of module, which can be confusing. Rest assured, however, we'll cover packages in the next section on npm.

CommonJS 1.1.1 specification can be found at <http://gswg.io#commonjs>.

1. module – an object representing the module itself. The module object contains the exports object. In the case of Node.js, it also contains meta-information, such as id, parent, and children.
2. exports – a plain JavaScript object, which may be augmented to expose functionality to other modules. The exports object is returned as the result of a call to require.
3. require – a function is used to import modules, returning the corresponding exports object.

The CommonJS specification also contains the following sample code, slightly modified for the purpose of clarity:

//Code example 01-modules

//program.js

var inc = **require**('./increment').increment;

var a = 1;

console.log(inc(a));

//increment.js

var add = **require**('./math').add;

**exports**.increment = function(b) {

return add(b, 1);

};

**//math.js**

**exports**.add = function(c, d) {

return c + d;

};

**$ node program.js**

**2**

**FINDING Modules**

**$ npm search gruntplugin rsync**

**npm install -g grunt-cli**

A Brief Tutorials about the Javascript Build Process

1. NodeJS
2. NPM
3. GRUNT
4. BOWER
5. Yoeman
6. GIT/GIT-SCM
7. SASS
8. Compass

Node.js (http://gswg.io#node) is a platform for writing JavaScript command-line tools, which run on all major operating systems.

Node.js brings JavaScript to the server and the desktop. While initially JavaScript was mainly used as a browser based language, with Node you can now also create your server-side backend or even a [desktop application](http://strongloop.com/strongblog/creating-desktop-applications-with-node-webkit/) with [node-webkit](https://github.com/rogerwang/node-webkit)

Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js can run the javascript program and using it ,it’s very easy to create Webserver.

# What is npm?

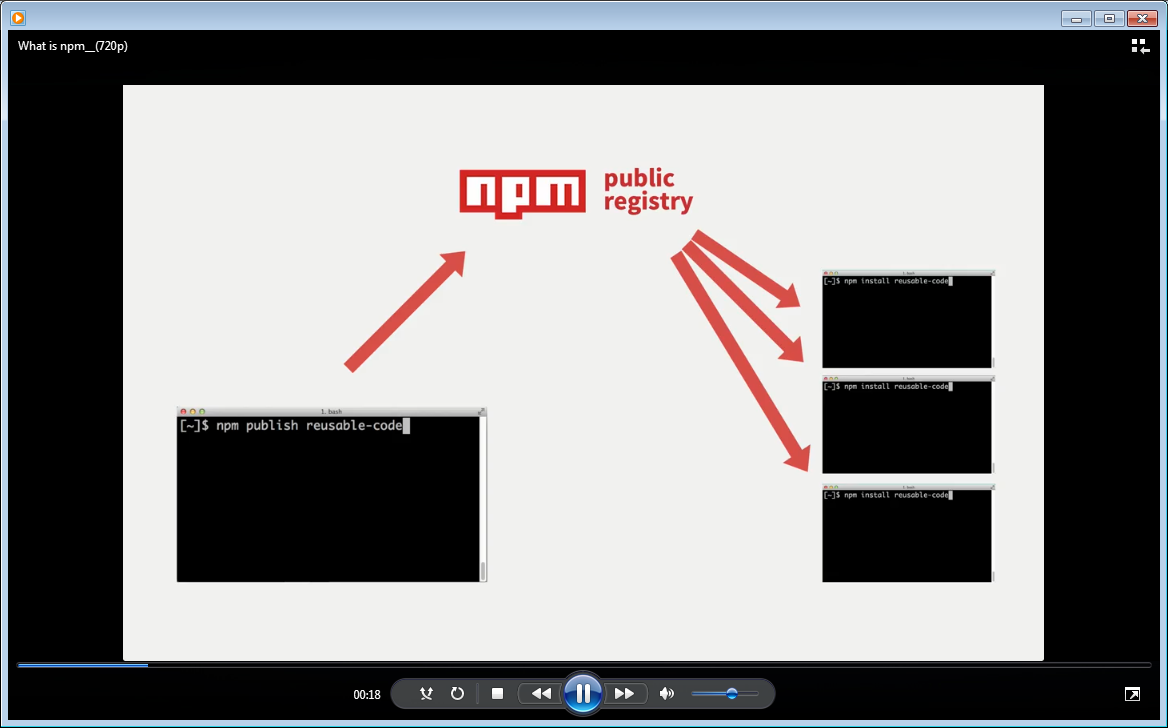
npm makes it easy for JavaScript developers to share and reuse code, and it makes it easy to update the code that you're sharing.

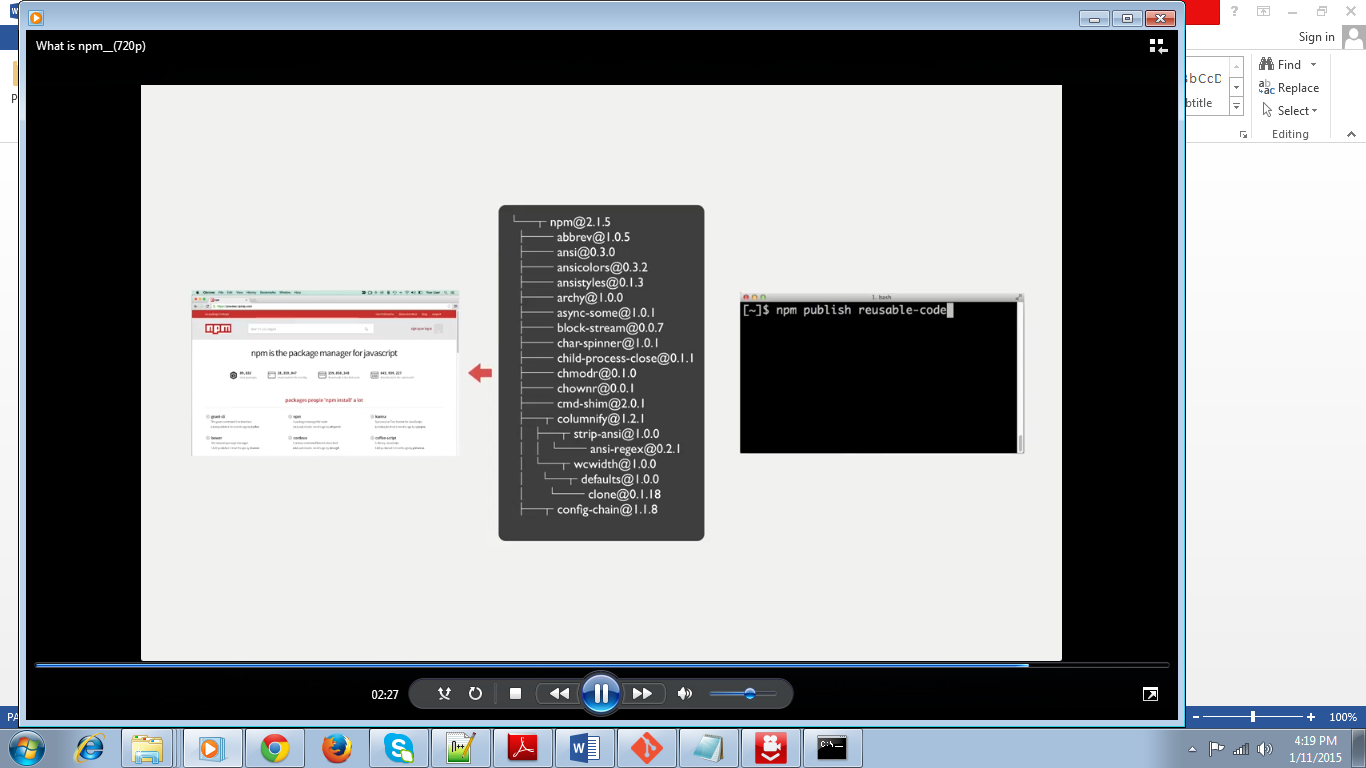
Getting npm to work behind a proxy requires setting the proxy andhttps-proxy settings. The key is noticing the - (dash) is not an \_ (underscore).

npm config set proxy http://proxy.company.com:8080

npm config set https-proxy <http://proxy.company.com:8080>

npm config set proxy http://"user:P@ssword"@proxy.server:1234





  
> node start.js

For More Information please visit:

<https://docs.npmjs.com/getting-started/what-is-npm>



Build tools or task runners are most commonly used for automating repetitive tasks, though we will see that the benefits of using Grunt far exceed simple automation.

Build Process may consist of following steps:

style and coding practice enforcement

compiling,

file watching

automatic task execution

unit testing

end-to-end testing

Grunt is run on NodeJS plateform. Once installed, we can execute grunt on the command line. This tells Grunt to look for a Gruntfile.js file. This choice of name refers to the build tool **Make**, and its Makefile. This file is the entry point to our build, which can define tasks inline, load tasks from external files, load tasks from external modules, and configure these tasks and much more.

//Code example 01-minify

module.exports = function(grunt) {

// Load the plugin that provides the "uglify" task.

grunt.loadNpmTasks('grunt-contrib-uglify');

// Project configuration.

grunt.initConfig({

uglify: {

target1: {

src: 'foo.js',

dest: 'foo.min.js'

}

}

});

// Define the default task

grunt.registerTask('default', ['uglify']);

};

What is need of Build Tool in javascript

Complex JavaScript Web Applications require considerable design and planning. It is quite common for the client-side (or browser) JavaScript code to be more complicated than the server-side code. With this in mind, we need to ensure our code base is manageable and maintainable.

1. *logically structure our project*
2. *keep our code DRY*
3. single source of truth

Grunt is a great tool for achieving these goals.

**Static analysis or Linting:**

In programming, the term **linting** is the process of finding *probable* bugs and/or style errors. Linting is more popular in dynamically typed languages as type errors may only be resolved at runtime. Douglas Crockford popularized JavaScript linting in 2011 with the release of his popular tool, JSLint.

**Transcompilation:**

Transcompiling—also known as source-to-source compilation and often abbreviated to transpiling—is the process of converting the source code of one language to the source code of another

Haml, Jade, Sass, LESS, Stylus, CoffeeScript, Dart, TypeScript, and more.

**Need to Compress the Web Files**

**Minification**

*73% of mobile Internet users say they have encountered a website that was too slow to load.*

*51% of mobile Internet users say they have encountered a website that crashed, froze, or received an error.*

*38% of mobile Internet users say they have encountered a website that wasn't available.*

*47% of consumers expect a web page to load in 2 seconds or less.*

40% of people abandon a website that takes more than 3 seconds to load.

1. Minify JavaScript—http://gswg.io#grunt-contrib-uglify
2. Minify CSS—http://gswg.io#grunt-contrib-cssmin
3. Minify HTML—http://gswg.io#grunt-contrib-htmlmin

**Concatenation**

As with minification, concatenation (or joining) also helps reduce page load time. As per the HTTP 1.1 specification, browsers can only request two files at once (see HTTP 1.1 Pipelining). Although newer browsers have broken this rule and will attempt to load up to six files at once, we will see it is still the cause of slower page load times.

**Deployment**

**ftp ,smtp**

BOWER:



Bower is a package manager for the web. If you’re not sure what package managers are, you’re gonna learn what they are and how useful they can be.

Installing Bower

To install bower, you need to have node.js installed on your computer. To check whether you have it or not, open up your command line interface (CLI) and type

$ node -v

If node.js is installed on your computer you should see the version you have. If not, then you can download it from [nodejs.org](http://nodejs.org/).

To install bower, just type

$ npm install -g bower

Using Bower

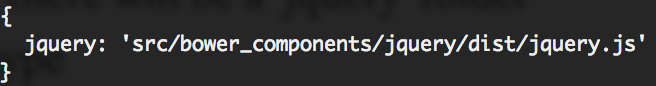
Create a new folder for your project and go to that folder in your CLI. Let’s say we want to add jQuery to our project. All you have to do is

$ bower install jquery

Now, if you open up the folder, you will see a new ‘src’ folder. Inside that folder should be a ‘bower\_components’ folder. All the packages you install will be inside that folder. If you open it up, there will be a ‘jquery’ folder with all the jquery code. Now type

$ bower list --paths

It will show you all the packages you installed along with their paths. However, sometimes it will only show the partial path. In that case you would have to go the folder and find the path.



Copy that path and put it as a source in a script tag inside your html file. Now you have jQuery in your project!

Is that it?

You’re probably asking yourself “is that all it does? I can just go grab the link for jQuery from their website and put it in my project.” But that’s definitely not only what Bower can do.

Bower Dependencies

Now what if you’re working in a group or you want to put your project on GitHub. It’s not a good idea to add all the source code to your repository. This is when Bower becomes very useful; it allows you to make a package file. To create a bower package file, type

$ bower init

It will ask you for a bunch of things like the name of the project, the description, etc… If you keep pressing the return key, it’s going to choose the defaults (that’s what I tend to do). Now if you open your project folder, you will see a new ‘bower.json’ file. So now, you don’t need to add the ‘bower\_components’ folder to your repository. To see how this works, delete the ‘bower\_components’ folder. Now type

$ bower install

If you did that, you will see that bower created a ‘bower\_components’ folder again with all the packages. If anyone wants to work on your project, that’s all they have to do.

After your ‘bower.json’ file has been created, you will need to add a ‘-S’ to the end of the package installation command so that it adds that package to the file. Let’s say we want to add bootstrap to our project

$ bower install bootstrap -S

This will add bootstrap to the ‘bower\_components’ folder and as a dependency in the ‘bower.json’ file. So remember, always add a ‘-S’ when installing new packages after creating a ‘bower.json’ file.

What else can Bower do?

Say you wanted a specific version of jQuery. Let’s see how you can do that. First, type

$ bower uninstall jquery -S

Which will remove the jQuery package from the ‘bower\_components’ folder as well as remove it as a dependancy from the ‘bower.json’ file. Let’s say we want jQuery version 1.9.1, all we have to do is

$ bower install jquery#1.9.1 -S

Done. See how simple that is? There’s no need to search for the specific version, download the zip file, and extract it into your folder.

Bower can also install git repositories

$ bower install git://github.com/user/package.git

and URLs

$ bower install http://example.com/script.js



Yeoman is build around [**generators**](http://yeoman.io/generators/) which are either developed by the Yeoman team official generators or by the open source community. Yeoman itself basically just provides the infrastructure for building and running those generators.

*Yeoman helps you kickstart new projects, prescribing best practices and tools to help you stay productive.*From the [official site](http://yeoman.io/)

What's nice about such approach is

* that you can quickly get up to speed. Creating a project setup with proper tools and dev support can cost you lots of time and requires expert knowledge.
* that you don't necessarly have to know all the best practices tools that are currently available on the market. Yeoman assembles them for you, s.t. you can get started immediately. Then once you get more expertise, you can adjust Yeoman's configuration to make it fit even more to your project needs.
* a great way for you to learn lots and lots of new tools.

Yeoman as well as its generators are distributed as a node modules. Simply install it globally

$ npm install -g yo

Then find your generator (i.e. [for angular](https://github.com/yeoman/generator-angular)) and install it using the following command.

$ npm install -g generator-angular

Finally, execute the generator within your project directory to create a new app.

$ yo angular [app-name]

This will create the initial scaffold from which you can then start building your application. But Yeoman goes even further, based on the generator you use, you may also generate single components, like Angular controllers, directives etc. while you develop.

$ yo angular:controller user