



## **npm ls**

Everything you have installed in the current directory.

## **npm search [search terms]**

Search the registry for packages matching the search.

## **npm install [<name> [<name> ...]] [--save|--save-dev|--save-optional]**

This command installs a package, or packages, and any packages that it depends on in the current directory. If the package has a shrinkwrap file, the installation of dependencies will be driven by that.

If no package.json exists, these options is ignored. If it exists, they'll update them if they are already there.

--save: Package will appear in your dependencies.

--save-dev: Package will appear in your devDependencies.

--save-optional: Package will appear in your optionalDependencies.

## **npm init**

Asks you a bunch of questions, and then writes a package.json for you. If you already have a package.json file, it'll read that first, and default to the options in there.

## **npm uninstall**

Uninstalls a package, completely removing everything npm installed on its behalf.

## **npm update [<name> [<name> ...]]**

This command will update all the packages listed to the latest version (specified by the tag config). It will also install missing packages.

## ABOUT THE AUTHOR

Jesse Warden, Software Consultant at Web App Solution. Contact us today!

[jessew@webappsolution.com](mailto:jessew@webappsolution.com) | (678) 253-7730

See our client list and work at <http://webappsolution.com>

Read Jesse's programming and fitness thoughts at <http://jessewarden.com>



## Bower

Front End Package Management, JavaScript's "Maven"

*Depends on Node, npm, and git*

Installing globally:

```
npm install -g bower
```

### **bower list**

Show all the packages that are installed locally.

### **bower search [<name>]**

Search for packages registered with Bower. Using just bower search will list all packages in the registry.

### **bower install**

Using the dependencies listed in the current directory's bower.json

### **bower install <package>**

Using a local or remote package.

### **bower uninstall <package>**

Uninstall a locally installed package.



# GRUNT

## Grunt

Automated Task Runner, JavaScript's "Ant"

*Depends on Node and npm*

Installing globally:

```
npm install -g grunt-cli
```

## grunt

Executes the Gruntfile.js located in the current directory. Specifically, the default task(s).

## grunt-init

Grunt-init is a scaffolding tool used to automate project creation. It will build an entire directory structure based on the current environment and the answers to a few questions. The exact files and contents created depend on the template chosen along with the answers to the questions asked.

Examples:

grunt-init-gruntfile: Create a basic Gruntfile.

grunt-init-jquery: Create a jQuery plugin, including QUnit unit tests.

grunt-init-commonjs: Create a commonjs module, including Nodeunit unit tests.

## Example in 5 Steps:

Step 1: package.json that contains:

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.1",
    "grunt-contrib-jshint": "~0.6.3",
    "grunt-contrib-nodeunit": "~0.2.0",
    "grunt-contrib-uglify": "~0.2.2"
  }
}
```

Step 2: Create a Gruntfile.js that contains:

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.to
```

### Step 1: package.json that contains:

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.1",
    "grunt-contrib-jshint": "~0.6.3",
    "grunt-contrib-nodeunit": "~0.2.0",
    "grunt-contrib-uglify": "~0.2.2"
  }
}
```

### Step 2: Create a Gruntfile.js that contains:

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.
today("yyyy-mm-dd") %> */\n'
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Load the plugin that provides the "uglify" task.
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).
  grunt.registerTask('default', ['uglify']);

}
```

### Step 3: Run “sudo npm install”

### Step 4: Create simple JavaScript file in a folder caled “src” and name it “simple-example.js”

### Step 5: Run “grunt”. Notice it’ll minify your JavaScript file in the build directory.

Step 1:  
package.json that contains:

# TypeScript Example

Use tomorrow's JavaScript, today for application-scale development.

*Depends on Node and npm*

Installing globally:

**npm install -g typescript**

Compile:

**tsc helloworld.ts**

**Step 1: package.json that contains:**

```
{
  "name": "helloworld",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.1",
    "typescript": "~0.8.3",
    "grunt-typescript": "~0.1.3",
    "grunt-contrib-watch": "~0.3.1",
    "grunt-contrib-connect": "~0.2.0",
    "grunt-open": "~0.2.0"
  }
}
```

**Step 2: Create a Gruntfile.js that contains:**

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    connect: {
      server: {
        options: {
          port: 8000,
          base: "./"
        }
      }
    },
    typescript: {
      base: {
        src: ["src/**/*.ts"],
        dest: "build/",
        options: {
          module: "amd",
          target: "es5"
        }
      }
    }
  });
};
```

```

    }
  },
  watch: {
    files: "src/**/*.ts",
    tasks: ["typescript"]
  },
  open: {
    dev: {
      path: "http://localhost:8000/index.html"
    }
  }
});

grunt.loadNpmTasks('grunt-typescript');
grunt.loadNpmTasks('grunt-contrib-watch');
grunt.loadNpmTasks('grunt-contrib-connect');
grunt.loadNpmTasks('grunt-open');

grunt.registerTask('default', ['connect', 'open', 'watch']);
};

```

### Step 3: Run “sudo npm install”

### Step 4: Create simple TypeScript file in a folder called “src” and name it “helloworld.ts”

```

import MessageModule = module("com/Message")

module Sayings {
  export class Greeter {
    message: MessageModule.com.Message;
    constructor(message: MessageModule.com.Message) {
      this.message = message;
    }
    greet() {
      return "Hello, " + this.message.greet();
    }
  }
}

var Message = MessageModule.com.Message;
var message = new Message("world");
var greeter = new Sayings.Greeter(message);

var button = document.createElement('button');
button.textContent = "Say Hello";
button.onclick = function() {
  alert(greeter.greet());
};

document.body.appendChild(button);

```

**Step 5: Create a folder called “com” inside of “src”.  
Create a file inside “com” called “Message.ts”:**

```
module com.jessewarden {  
  export class Greeter {  
    greeting: string;  
    constructor(message: string) {  
      this.greeting = message;  
    }  
    greet() {  
      return "Hello, " + this.greeting;  
    }  
  }  
}
```

**Notice your generated code is setup as a RequireJS AMD module.**