



This set of slides illustrate the steps for installing Node.js and Express on Windows. **Please don't print it in order to save paper!**

## CSCI 4140 – Tutorial 5

# Installing Node.js and Express on Windows

Matt YIU, Man Tung ([mtyiucse](mailto:mtyiucse))

SHB 118

*Office Hour: Tuesday, 3-5 pm*

2015.02.12



# Installing Node.js on Windows

*Next, next, next, accept, install... FINISH*

# Step 1: Download the Windows installer

- Download the latest version of Node.js from <http://nodejs.org/download/>
- Most of you should be using 64-bit machine already 😊

## Downloads

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Current version: v0.10.35



**Windows Installer**

node-v0.10.35-x86.msi



**Macintosh Installer**

node-v0.10.35.pkg



**Source Code**

node-v0.10.35.tar.gz

Windows Installer (.msi)

32-bit

64-bit

Windows Binary (.exe)

32-bit

64-bit

Mac OS X Installer (.pkg)

Universal

Mac OS X Binaries (.tar.gz)

32-bit

64-bit

Linux Binaries (.tar.gz)

32-bit

64-bit

SunOS Binaries (.tar.gz)

32-bit

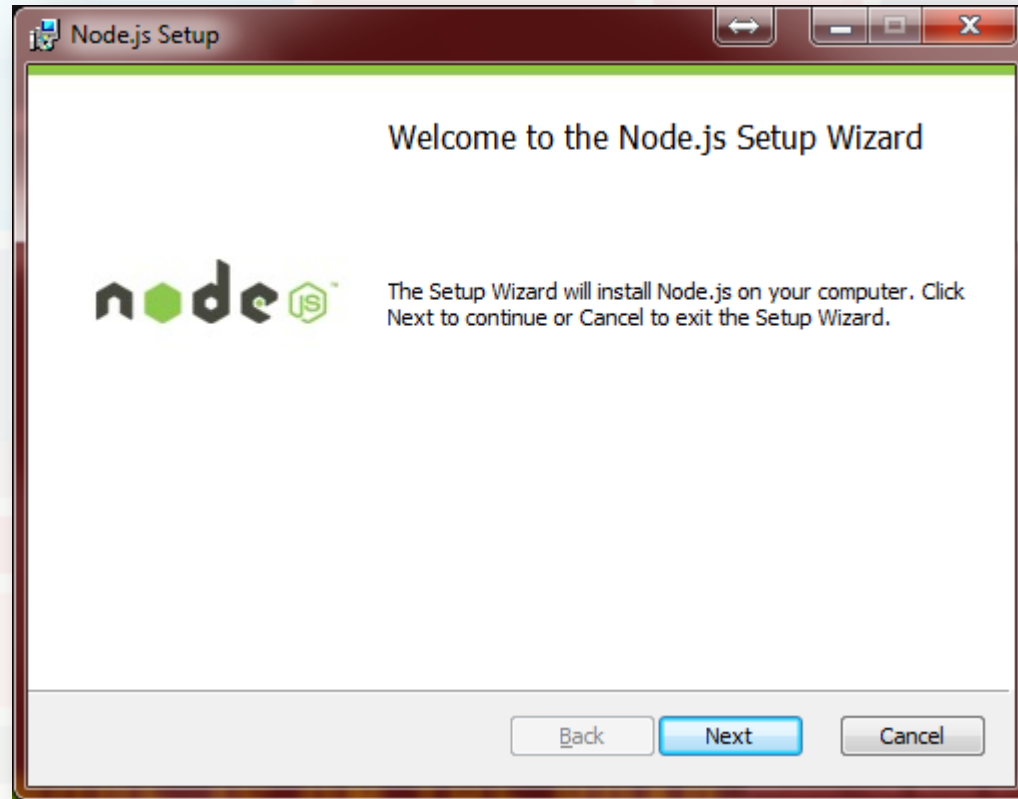
64-bit

Source Code

node-v0.10.35.tar.gz

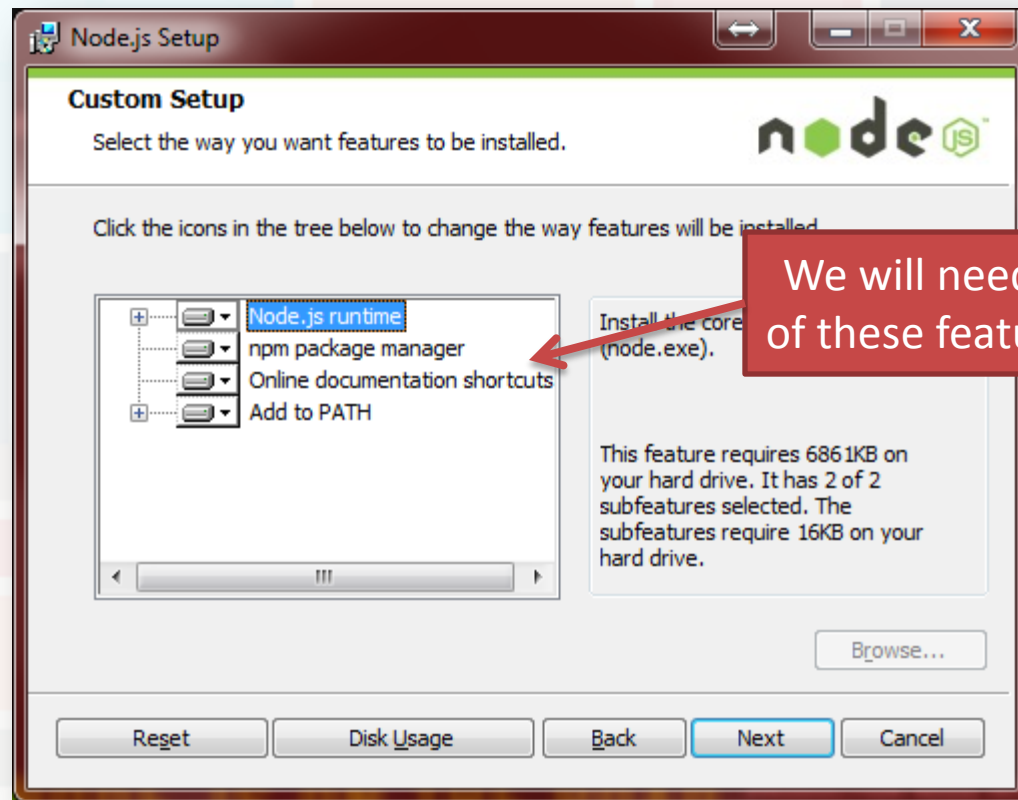
## Step 2: Install Node.js

- Execute the installer...*Next, next, ...*



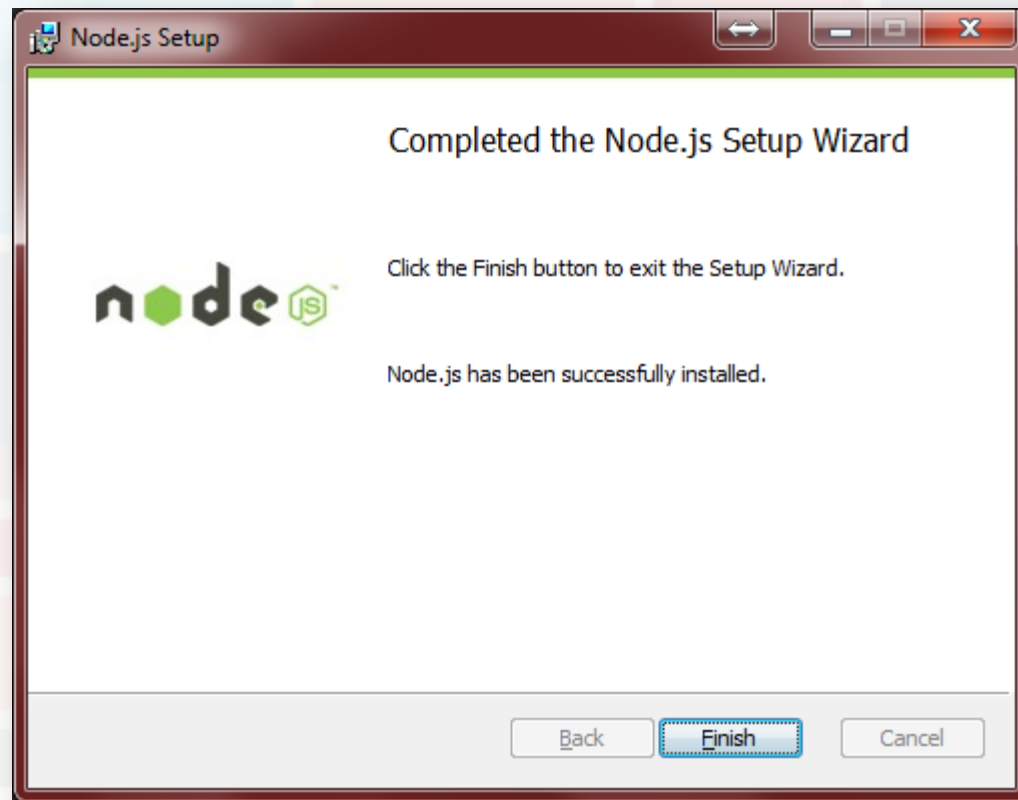
## Step 2: Install Node.js

- Execute the installer...*Next, next, ...*



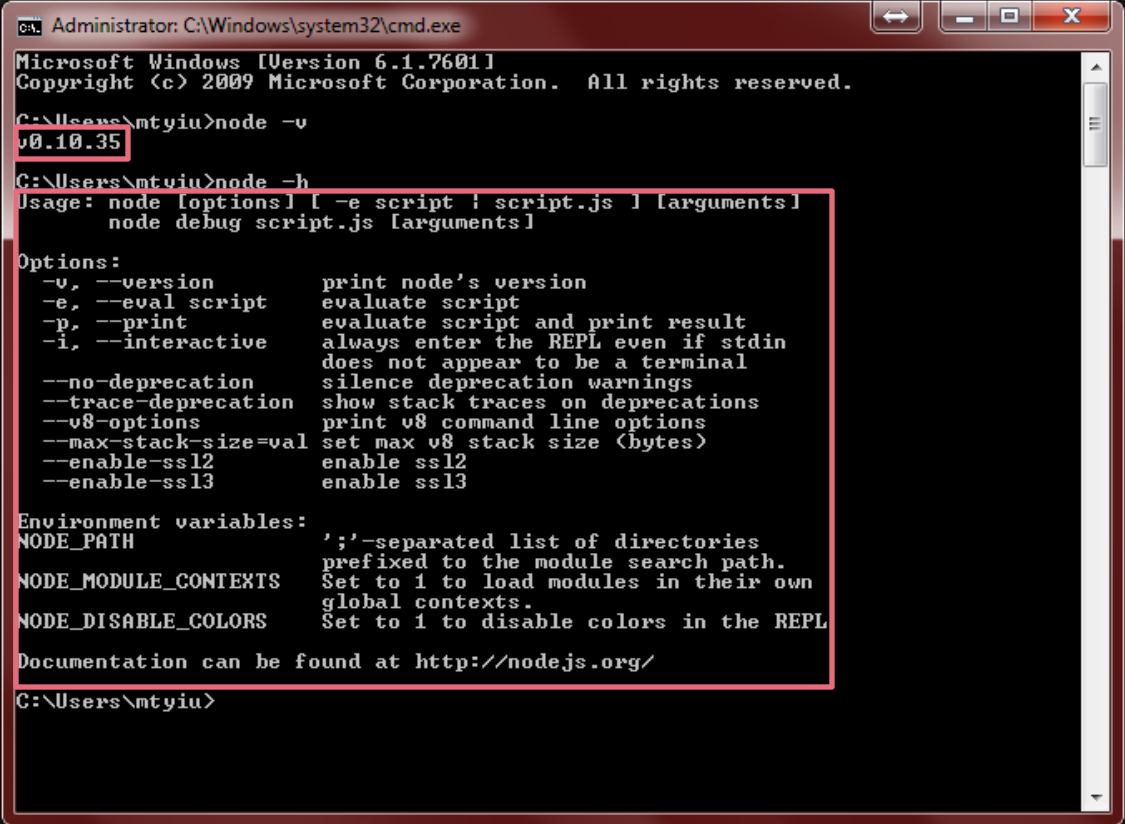
## Step 2: Install Node.js

- Finish!



## Step 3: Test your Node.js installation

- Open your command prompt:
  - Windows Key + R → Type “cmd”
- Enter “**node -v**” to display the version number of your Node.js installation
- Enter “**node -h**” to display the help message of Node.js



```
C:\Users\mtviu>node -v
v0.10.35
C:\Users\mtviu>node -h
Usage: node [options] [-e script | script.js | arguments]
       node debug script.js [arguments]

Options:
  -v, --version           print node's version
  -e, --eval script       evaluate script
  -p, --print             evaluate script and print result
  -i, --interactive       always enter the REPL even if stdin
                          does not appear to be a terminal
  --no-deprecation        silence deprecation warnings
  --trace-deprecation     show stack traces on deprecations
  --v8-options            print v8 command line options
  --max-stack-size=val   set max v8 stack size (bytes)
  --enable-ssl2           enable ssl2
  --enable-ssl3           enable ssl3

Environment variables:
NODE_PATH                ';' separated list of directories
                          prefixed to the module search path.
NODE_MODULE_CONTEXTS     Set to 1 to load modules in their own
                          global contexts.
NODE_DISABLE_COLORS     Set to 1 to disable colors in the REPL

Documentation can be found at http://nodejs.org/
C:\Users\mtviu>
```

## Step 4: “Hello World”!

- Time to write our first Node.js program!

```
var http = require( 'http' );
http.createServer( function( request, response ) {
  response.writeHead( 200, { 'Content-Type' : 'text/plain' } );
  response.end( 'Hello World!\n' );
} ).listen( 4140, '127.0.0.1' );

console.log( 'Server running at http://127.0.0.1:4140/' );
```

**hello.js**

- Save the program anywhere you like
  - In this example, the file is saved under “D:\csci4140”



## Step 5: Say “Hello World” to the World!

- Get back to your command prompt again...
- Change the current directory to where `hello.js` is saved
- Execute `“node hello.js”` (simple enough?)

```
Administrator: C:\Windows\system32\cmd.exe - no
C:\Users\mtyu>:
D:\>cd csci4140
D:\csci4140>dir
Volume in drive D has no label.
Volume Serial Number is 864E-7554

Directory of D:\csci4140

01/22/2015  05:10 PM    <DIR>          .
01/22/2015  05:10 PM    <DIR>          ..
01/22/2015  05:09 PM                305 hello.js
               1 File(s)                305 bytes
               2 Dir(s)  60,143,370,240 bytes free

D:\csci4140>more hello.js
var http = require( 'http' );
http.createServer( function( request, response ) {
  response.writeHead( 200, { 'Content-Type' : 'text/plain' } );
  response.end( 'Hello World!\n' );

  console.log( request );
} ).listen( 4140, '127.0.0.1' );

console.log( 'Server running at http://127.0.0.1:4140/' );
D:\csci4140>node hello.js
Server running at http://127.0.0.1:4140/
```

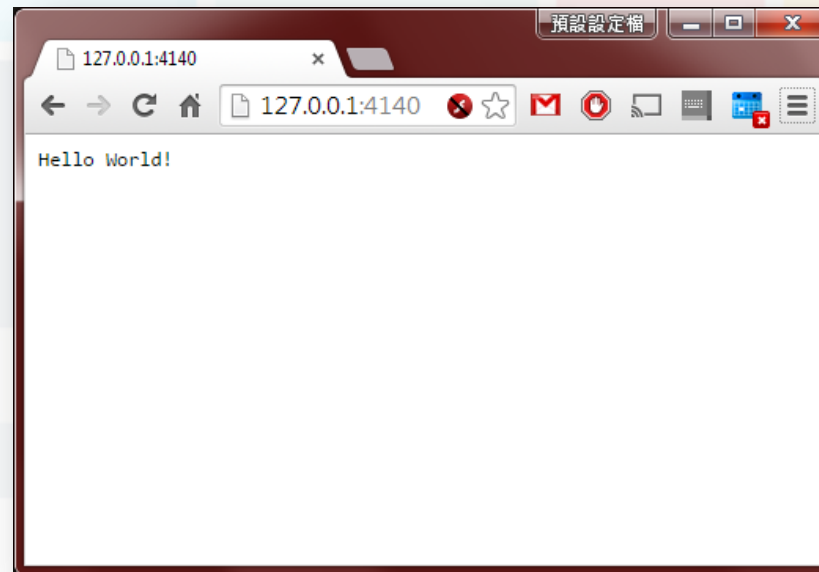
Changing current directory...

- Change to D drive: Enter “d:”
- List contents: “dir”
- Change directory: “cd [DIR\_NAME]”

Run!

## Step 5: Say “Hello World” to the World!

- Your first Node.js program is ready to test! Now use your browser to visit: <http://127.0.0.1:4140/>
- Can you see the result?



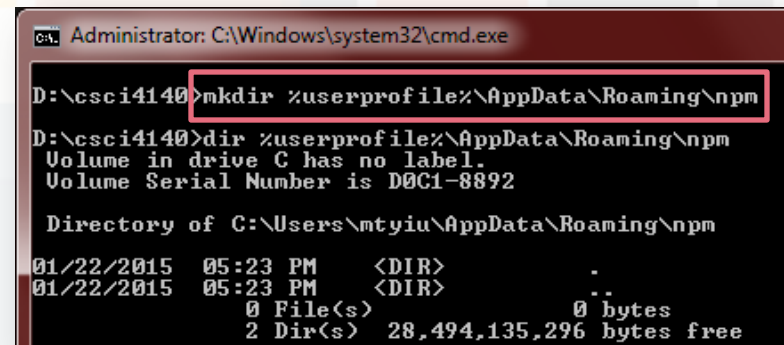
The background features a stylized illustration of a laptop and a tablet. The laptop screen is filled with several overlapping, semi-transparent windows in various colors: light blue, light orange, light red, light yellow, and light grey. The tablet also displays a similar arrangement of colorful windows. The overall aesthetic is clean and modern, typical of a technical presentation.

# Installing Express on Windows

*We will use npm package manager to install the Node.js framework.*

# Step 1. Initialize npm

- Don't ask me why...the Node.js installer does not create a folder for npm...
  - If you execute “npm” in the command prompt, you may get this error:  
Error: ENOENT, stat 'C:\Users\[Username]\AppData\Roaming\npm'
- To solve this problem, create the directory at the displayed path in command prompt:  
“**mkdir %userprofile%\AppData\Roaming\npm**”
  - You may need to run the command prompt as an administrator

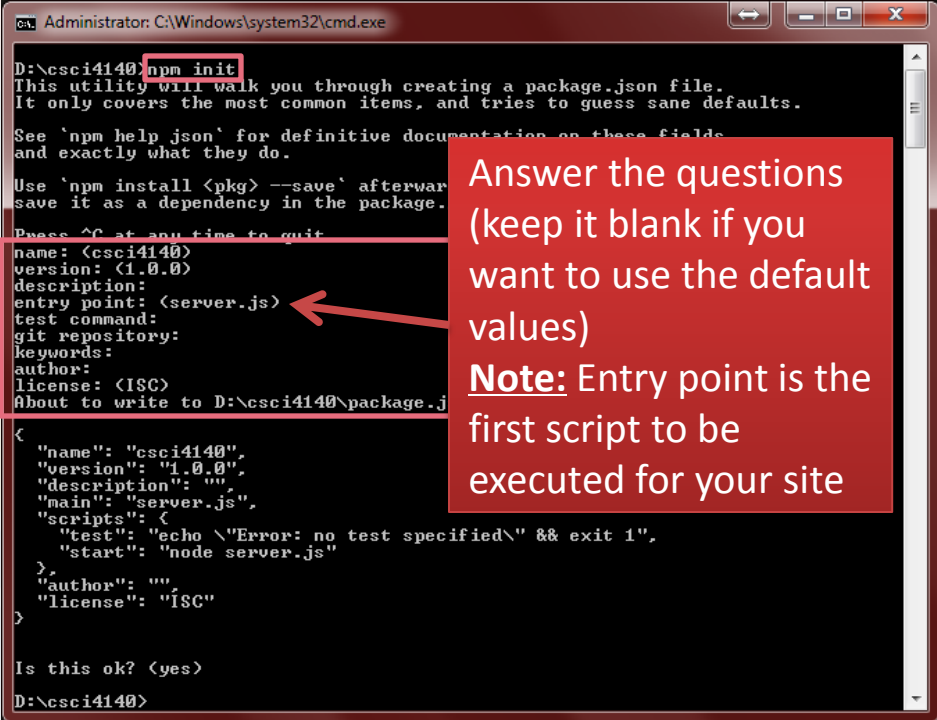


```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>mkdir %userprofile%\AppData\Roaming\npm
D:\csci4140>dir %userprofile%\AppData\Roaming\npm
Volume in drive C has no label.
Volume Serial Number is D0C1-8892

Directory of C:\Users\ntyiu\AppData\Roaming\npm
01/22/2015  05:23 PM    <DIR>          .
01/22/2015  05:23 PM    <DIR>          ..
                0 File(s)              0 bytes
                2 Dir(s)      28,494,135,296 bytes free
```

## Step 2. Create a package.json file

- Go to your project folder.  
We are going to create package.json for our new project with npm
  - package.json holds various **metadata** relevant to the project
  - It allows npm (Node.js package manager) to **identify the project** as well as handle the **project's dependencies**
- Execute “**npm init**”



```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sane defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards
save it as a dependency in the package.

Press ^C at any time to quit.
name: (csci4140)
version: (1.0.0)
description:
entry point: (server.js)
test command:
git repository:
keywords:
author:
license: (ISC)
about to write to D:\csci4140\package.j

{
  "name": "csci4140",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC"
}

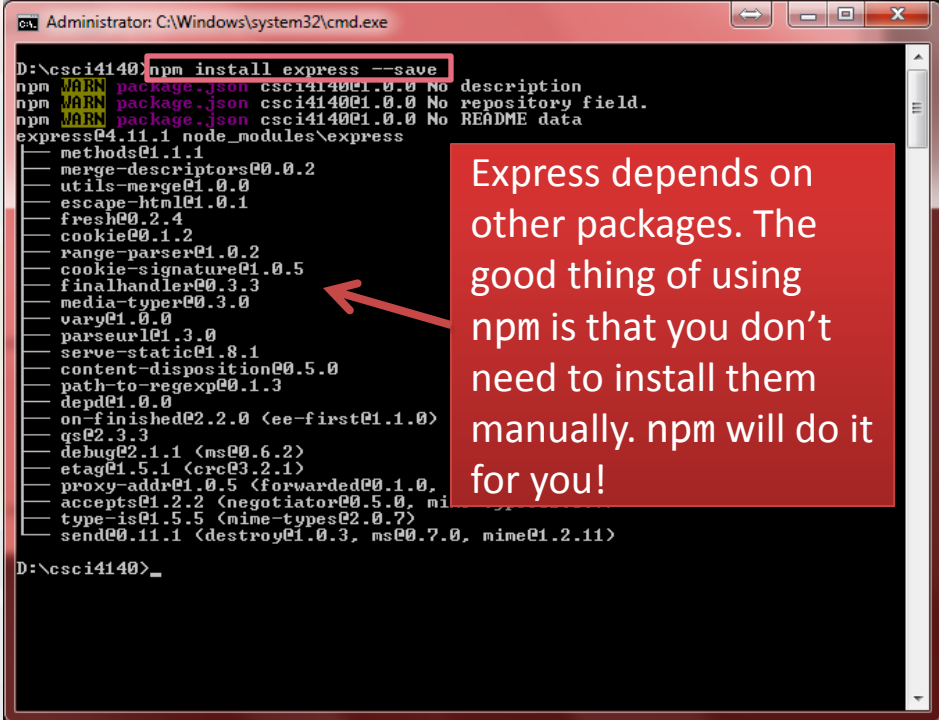
Is this ok? (yes)
D:\csci4140>
```

Answer the questions  
(keep it blank if you  
want to use the default  
values)

**Note:** Entry point is the  
first script to be  
executed for your site

## Step 3. Install Express

- We are ready to install Express now
  - Express is a “*Fast, unopinionated, minimalist web framework for Node.js*”
  - It is useful for building web applications
- Execute “**npm install express --save**”
  - This installs Express in the app directory and save it in the dependencies list

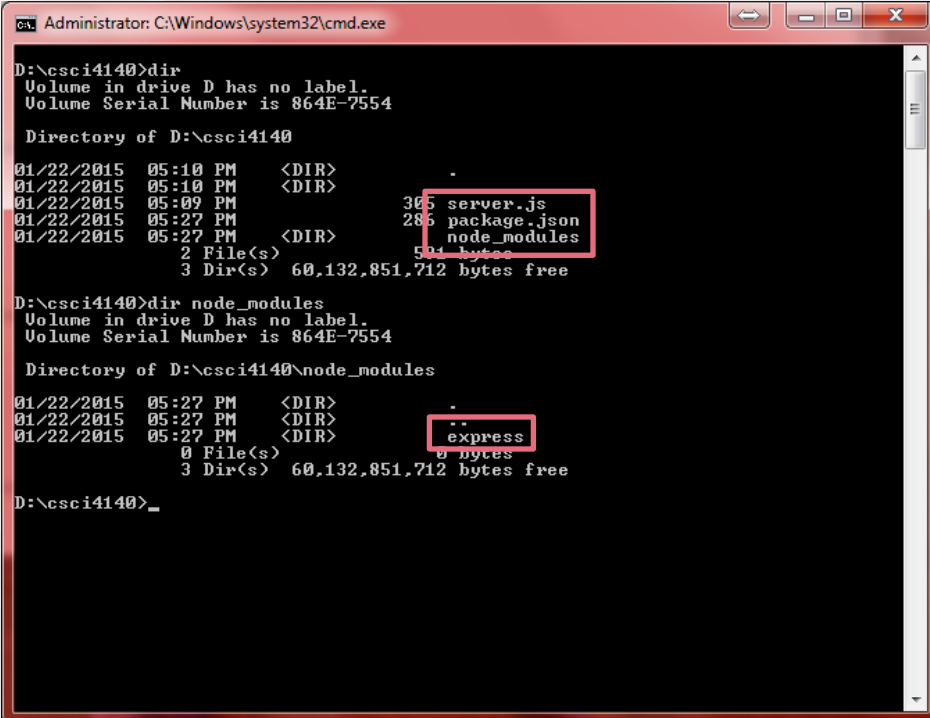


```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>npm install express --save
npm WARN package.json csci4140@1.0.0 No description
npm WARN package.json csci4140@1.0.0 No repository field.
npm WARN package.json csci4140@1.0.0 No README data
express@4.11.1 node_modules\express
  methods@1.1.1
  merge-descriptors@0.0.2
  utils-merge@1.0.0
  escape-html@1.0.1
  fresh@0.2.4
  cookie@0.1.2
  range-parser@1.0.2
  cookie-signature@1.0.5
  finalhandler@0.3.3
  media-typer@0.3.0
  vary@1.0.0
  parseurl@1.3.0
  serve-static@1.8.1
  content-disposition@0.5.0
  path-to-regexp@1.3
  depd@1.0.0
  on-finished@2.2.0 <ee-first@1.1.0>
  qs@2.3.3
  debug@2.1.1 <ms@0.6.2>
  etag@1.5.1 <crc@3.2.1>
  proxy-addr@1.0.5 <forwarded@0.1.0,
  accepts@1.2.2 <negotiator@0.5.0, mi
  type-is@1.5.5 <mime-types@2.0.7>
  send@0.11.1 <destroy@1.0.3, ms@0.7.0, mime@1.2.11>
D:\csci4140>
```

Express depends on other packages. The good thing of using npm is that you don't need to install them manually. npm will do it for you!

## Step 3. Install Express

- Check your installation. There should be a new directory called “node\_modules”
- Inside “node\_modules”, a directory called “express” is created



```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>dir
Volume in drive D has no label.
Volume Serial Number is 864E-7554

Directory of D:\csci4140

01/22/2015  05:10 PM  <DIR>          .
01/22/2015  05:10 PM  <DIR>          ..
01/22/2015  05:09 PM          305 server.js
01/22/2015  05:27 PM          285 package.json
01/22/2015  05:27 PM  <DIR>          node_modules
                2 File(s)          591 bytes
                3 Dir(s)  60,132,851,712 bytes free

D:\csci4140>dir node_modules
Volume in drive D has no label.
Volume Serial Number is 864E-7554

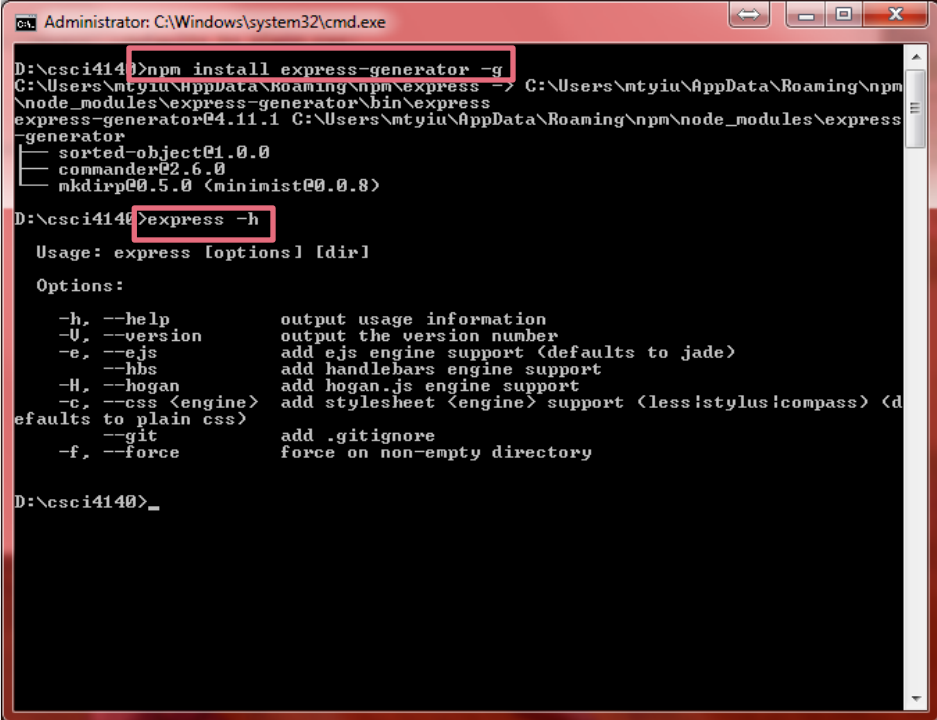
Directory of D:\csci4140\node_modules

01/22/2015  05:27 PM  <DIR>          .
01/22/2015  05:27 PM  <DIR>          ..
01/22/2015  05:27 PM  <DIR>          express
                0 File(s)          0 bytes
                3 Dir(s)  60,132,851,712 bytes free

D:\csci4140>_
```

## Step 4. Install Express application generator

- Next, we will install Express application generator
  - It is used to quickly create a Express application skeleton
  - This saves your work from defining the structure yourself!
- Execute “**npm install express-generator -g**”
- After installation, execute “**express -h**” to check your installation



```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>npm install express-generator -g
C:\Users\mtyiu\AppData\Roaming\npm\express -> C:\Users\mtyiu\AppData\Roaming\npm\node_modules\express-generator\bin\express-generator
express-generator@4.11.1 C:\Users\mtyiu\AppData\Roaming\npm\node_modules\express-generator
├── sorted-object@1.0.0
├── commander@2.6.0
└── mkdirp@0.5.0 <minimist@0.0.8>

D:\csci4140>express -h

Usage: express [options] [dir]

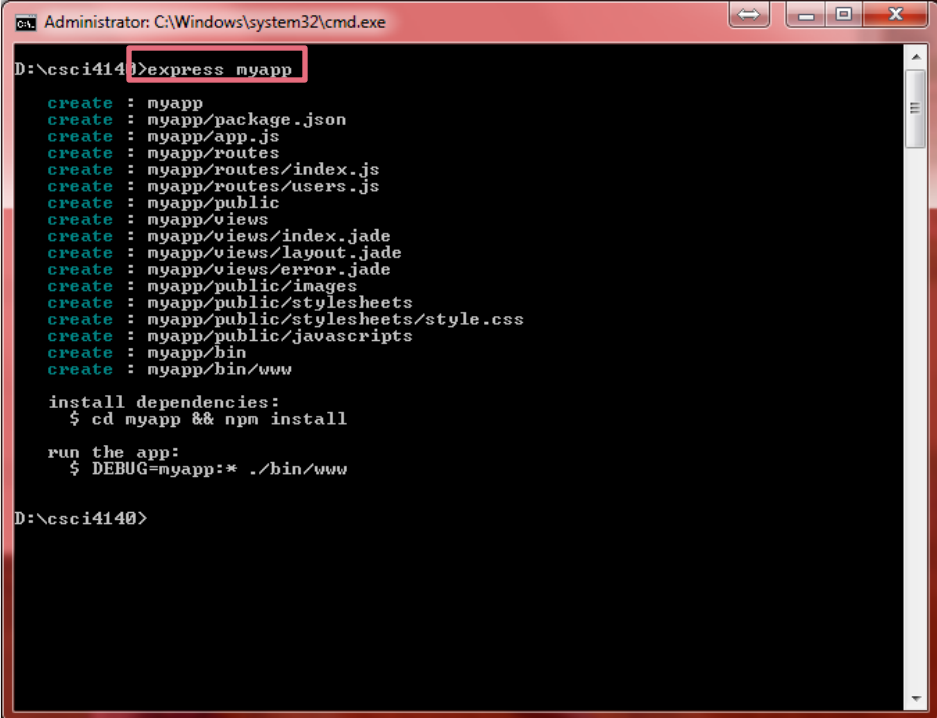
Options:
  -h, --help            output usage information
  -V, --version         output the version number
  -e, --ejs             add ejs engine support (defaults to jade)
  -H, --hbs             add handlebars engine support
  -c, --css <engine>  add hogan.js engine support
                       add stylesheet <engine> support (less!stylus!compass) <defaults to plain css>
  -f, --force          add .gitignore
                       force on non-empty directory

D:\csci4140>_
```



## Step 5. Create an Express app

- Use the generator to create our first Express app (let's call it myapp)
- Execute “**express myapp**”
  - Files are created under the directory “myapp”



```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>express myapp

create : myapp
create : myapp/package.json
create : myapp/app.js
create : myapp/routes
create : myapp/routes/index.js
create : myapp/routes/users.js
create : myapp/public
create : myapp/views
create : myapp/views/index.jade
create : myapp/views/layout.jade
create : myapp/views/error.jade
create : myapp/public/images
create : myapp/public/stylesheets
create : myapp/public/stylesheets/style.css
create : myapp/public/javascripts
create : myapp/bin
create : myapp/bin/www

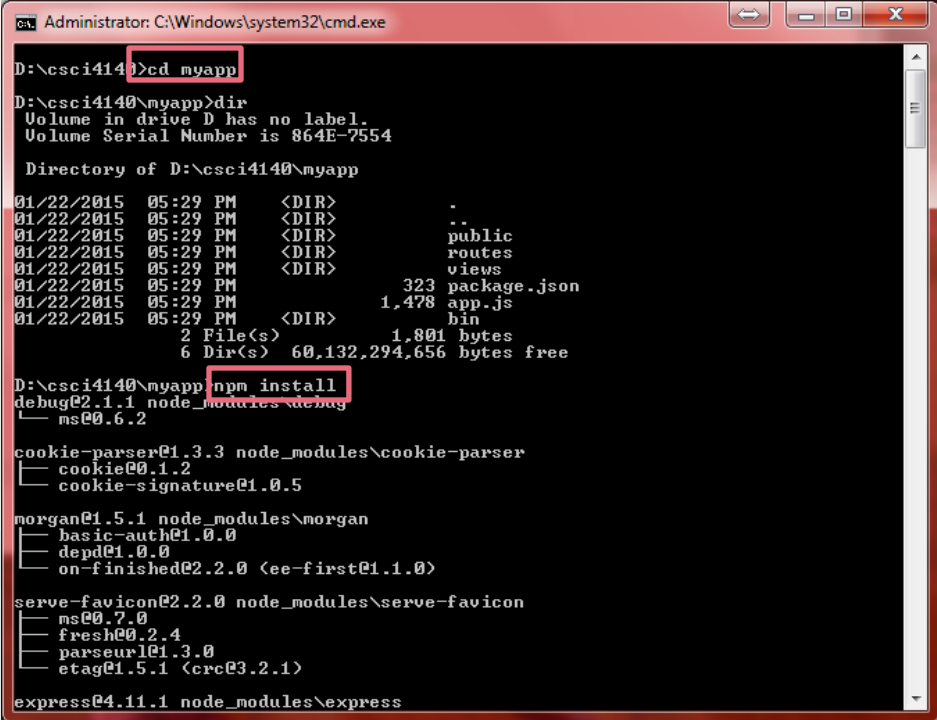
install dependencies:
$ cd myapp && npm install

run the app:
$ DEBUG=myapp:* ./bin/www

D:\csci4140>
```

## Step 6. Install dependencies

- Change the current directory to myapp with “**cd myapp**”
- Install dependencies with “**npm install**”



```
Administrator: C:\Windows\system32\cmd.exe
D:\csci4140>cd myapp
D:\csci4140\myapp>dir
Volume in drive D has no label.
Volume Serial Number is 864E-7554

Directory of D:\csci4140\myapp

01/22/2015  05:29 PM  <DIR>          .
01/22/2015  05:29 PM  <DIR>          ..
01/22/2015  05:29 PM  <DIR>          public
01/22/2015  05:29 PM  <DIR>          routes
01/22/2015  05:29 PM  <DIR>          views
01/22/2015  05:29 PM                323 package.json
01/22/2015  05:29 PM                1,478 app.js
01/22/2015  05:29 PM  <DIR>          bin
                2 File(s)      1,801 bytes
                6 Dir(s)  60,132,294,656 bytes free

D:\csci4140\myapp>npm install
debug@2.1.1 node_modules\debug
├─ ms@0.6.2

cookie-parser@1.3.3 node_modules\cookie-parser
├─ cookie@0.1.2
├─ cookie-signature@1.0.5

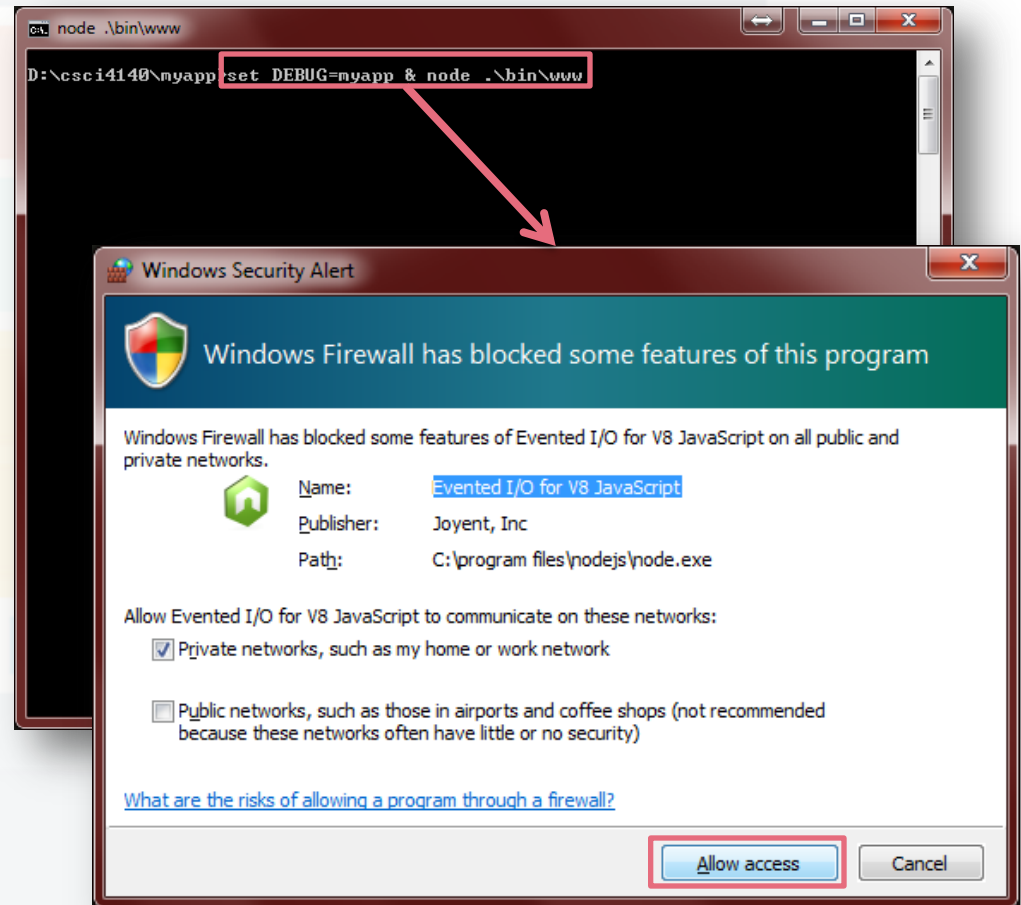
morgan@1.5.1 node_modules\morgan
├─ basic-auth@1.0.0
├─ depd@1.0.0
├─ on-finished@2.2.0 (ee-first@1.1.0)

serve-favicon@2.2.0 node_modules\serve-favicon
├─ ms@0.7.0
├─ fresh@0.2.4
├─ parseurl@1.3.0
├─ etag@1.5.1 (crc@3.2.1)

express@4.11.1 node_modules\express
```

## Step 7. Run the app

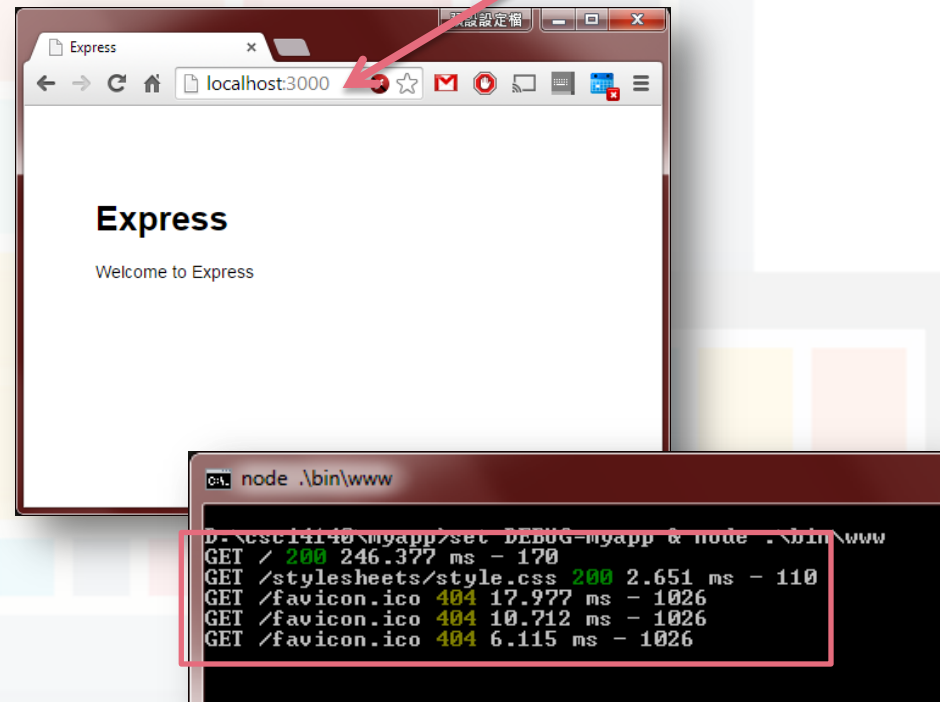
- Let's run the app to see what has been created
- Execute **“set DEBUG=myapp & node .\bin\www”**
- If you encounter a Windows Security Alert, press **“Allow access”**



## Step 7. Run the app

- Use your browser to visit <http://127.0.0.1:3000/>
  - The port number used by default is 3000
  - Of course, it is possible to change it
- At the same time, the command prompt will show some debug messages

**Note:** localhost is equivalent to 127.0.0.1



# Congratulations!

- You installed a development environment for Node.js on your Windows machine
- Please refer to the notes for deploying your Node.js applications to OpenShift

– End –