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

CSCI 4140 – Tutorial 5

Installing Node.js and Express on Linux or Mac

Matt YIU, Man Tung (mtyiu@cse)

SHB 118

Office Hour: Tuesday, 3-5 pm

2015.02.12

Installing Node.js on Linux with a package manager

E.g., Using "`apt-get install`" in Ubuntu

Linux

Install Node.js with a package manager

- Open your terminal
- If you are using Ubuntu, execute:

```
$ curl -sL https://deb.nodesource.com/setup | sudo bash -
$ sudo apt-get install -y nodejs npm
```

- If you are using other Linux distributions, see https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager for the installation instructions
- **Note:** The command for executing Node.js is "**nodejs**" instead of "***node***"!
  - To be able to invoke it by "**node**", execute "**`sudo ln -s /usr/bin/nodejs /usr/local/bin/node`**"

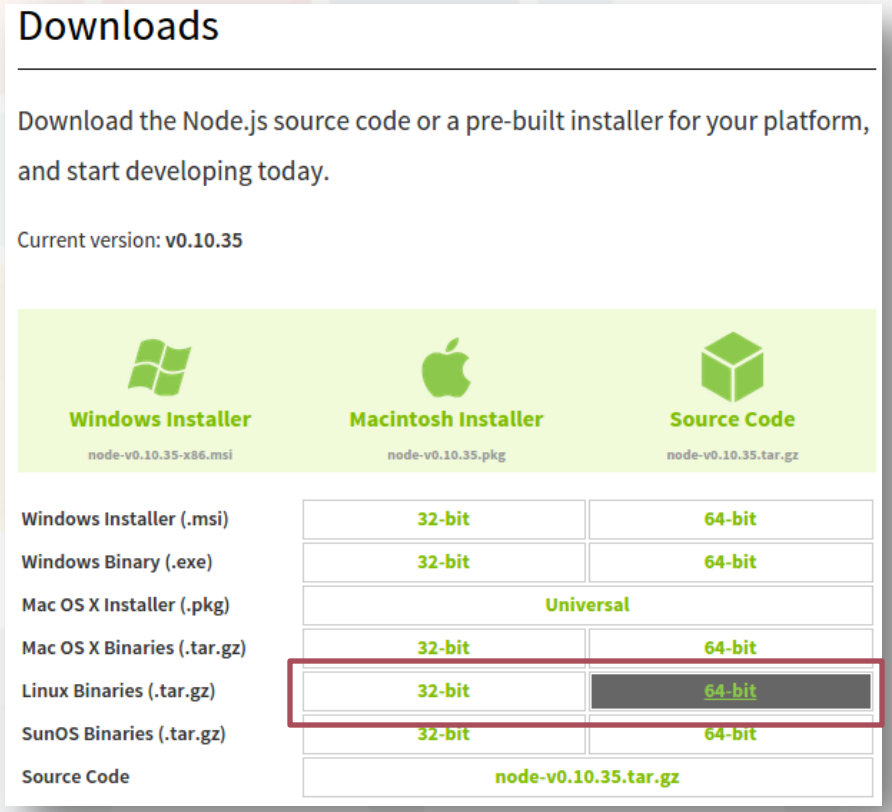
Installing Node.js on Linux without a package manager

This installation guide also applies to department's Linux machines.

Linux

Step 1. Download the Linux binaries

- Download the latest version of Node.js from http://nodejs.org/download/
- Most of you should be using 64-bit machine already ☺
  - If you are using a 32-bit machine, please remember to choose the 32-bit version



Linux

Step 2. Set up Node.js

- In your terminal, "**cd**" to the directory where the tarball is located and untar it

```
$ tar xvf node-v0.10.35-linux-x64
$ cd node-v0.10.35-linux-x64/bin
$ pwd
/home/mtyiu/csci4140/node-v0.10.35-linux-x64/bin
```

- Append the following lines to **~/.bashrc** (for bash shell) / **~/.cshrc** (for C shell, e.g., CSE department's Linux machines)

```
PATH=/home/mtyiu/csci4140/node-v0.10.35-linux-x64:$PATH
```

For ~/.bashrc

```
set path=($path /home/mtyiu/csci4140/node-v0.10.35-linux-x64/bin)
```

For ~/.cshrc

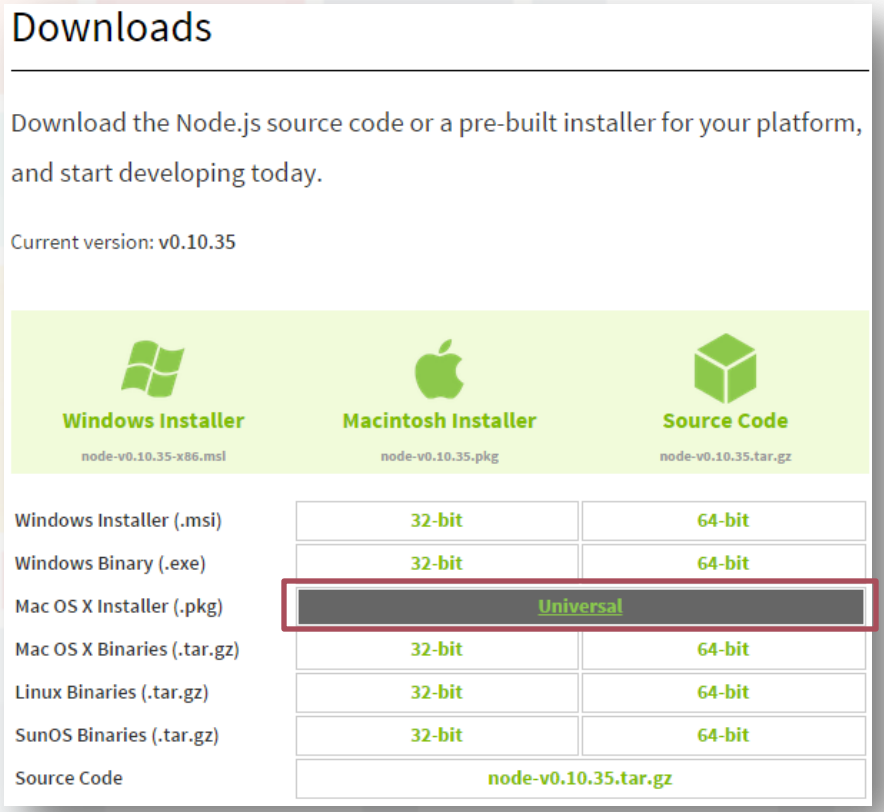
Installing Node.js on Mac

Node.js provides a convenient Mac OS X Installer!

Mac OS X

Download and execute the installer

- Download the latest version of Node.js from http://nodejs.org/download/
- No need to distinguish among 32-bit and 64-bit ☺
- Execute the .pkg file and follow the instructions



Testing your Node.js installation

To make sure that everything works properly…

Step 1: Test your Node.js installation

- Restart your terminal
- Enter "**node -v**" to display the version number of your Node.js installation
- Enter "**node -h**" to display the help message of Node.js

```
$ node -v
v0.10.35
$ 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/
$
```

Step 2: "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 "~/csci4140"

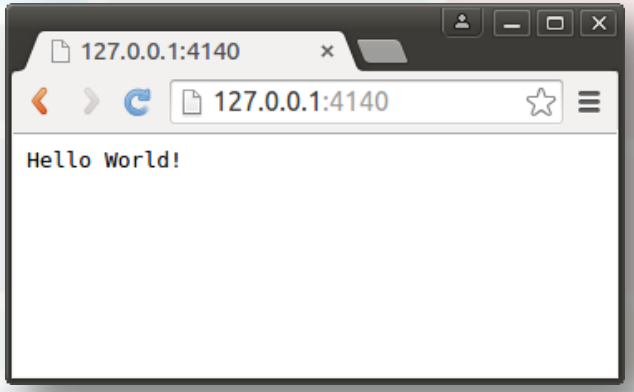
Step 3: Say "Hello World" to the World!

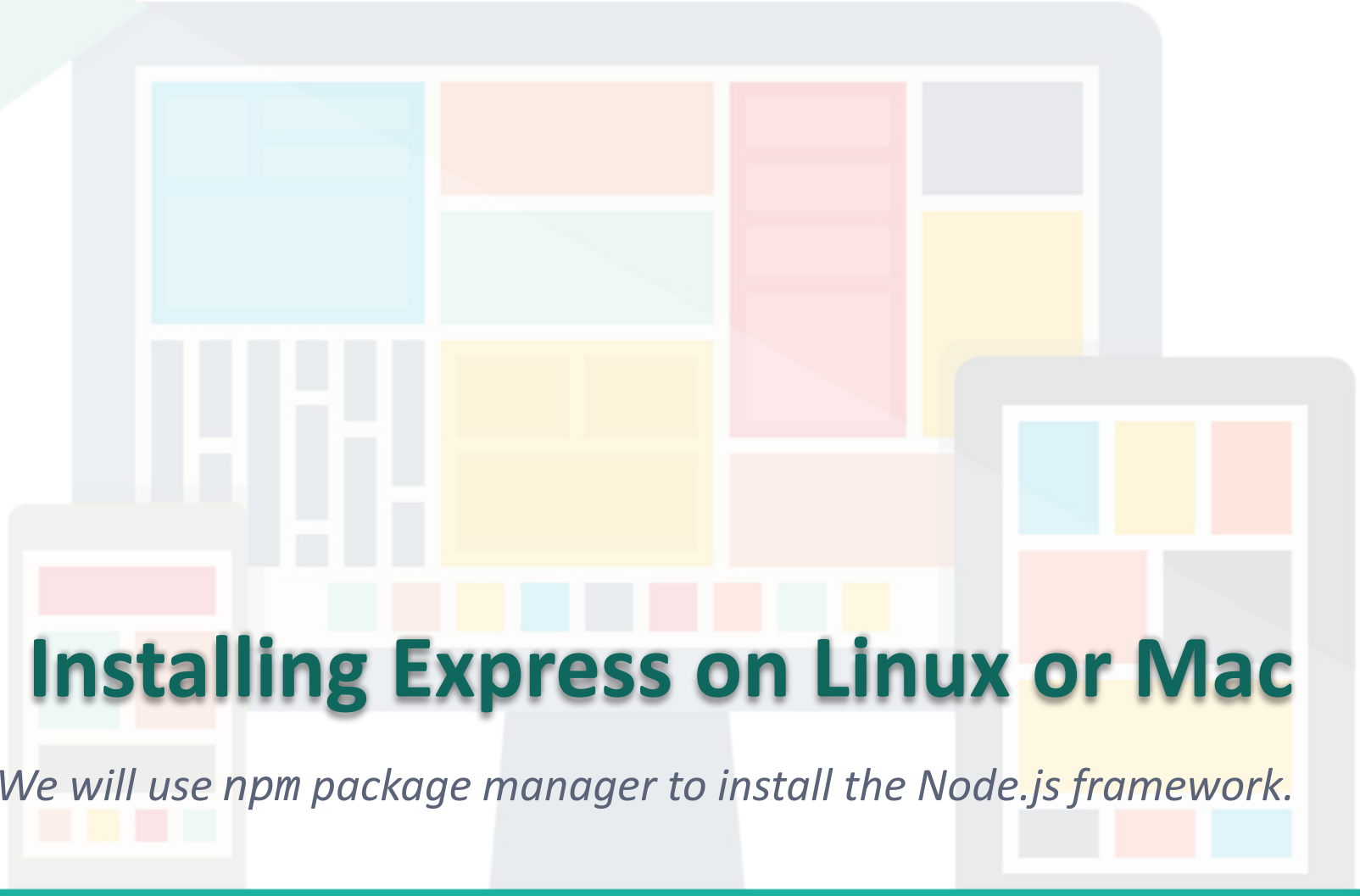
- Get back to your terminal again…
- Change the current directory to where `hello.js` is saved
- Execute "**`node hello.js`**" (simple enough?)

```
$ cd ~/csci4140/
$ node hello.js
Server running at http://127.0.0.1:4140/
```

Step 4: 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?





Installing Express on Linux or Mac

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

Step 1. 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**"

```
$ 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 to install a package
and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (csci4140)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/mtyiu/cs

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


Is this ok? (yes)

$
```

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 2. 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

```
$ 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
├── utils-merge@1.0.0
├── merge-descriptors@0.0.2
├── cookie@0.1.2
├── fresh@0.2.4
├── methods@1.1.1
├── escape-html@1.0.1
├── 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@0.1.3
├── depd@1.0.0
├── qs@2.3.3
├── on-finished@2.2.0 (ee-first@1.1.0)
├── debug@2.1.1 (ms@0.6.2)
├── proxy-addr@1.0.5 (forwarded@0.1.0, ipaddr.js@0.1.6)
├── send@0.11.1 (destroy@1.0.3, ms@0.7.0, mime@1.2.11)
├── etag@1.5.1 (crc@3.2.1)
├── accepts@1.2.2 (negotiator@0.5.0, mime-types@2.0.7)
└── type-is@1.5.5 (mime-types@2.0.7)
$
```

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

```
$ ls
./  ../  node_modules/  package.json
$ ls node_modules/
./  ../  express/
```

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

Add "**sudo**" if it failed

```
$ npm install express-generator -g
/home/mtyiu/csci4140/node-v0.10.35-linux-x64/bin/express ->
/home/mtyiu/csci4140/node-v0.10.35-linux-
x64/lib/node_modules/express-generator/bin/express
express-generator@4.11.1 /home/mtyiu/csci4140/node-
v0.10.35-linux-x64/lib/node_modules/express-generator
├── sorted-object@1.0.0
├── commander@2.6.0
└── mkdirp@0.5.0 (minimist@0.0.8)
$ 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)
        --hbs           add handlebars engine support
    -H, --hogan         add hogan.js engine support
    -c, --css <engine>  add stylesheet <engine> support
(less|stylus|compass) (defaults to plain css)
        --git           add .gitignore
    -f, --force         force on non-empty directory

$
```

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`"

```
$ express myapp

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

   install dependencies:
     $ cd myapp && npm install

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

$
```

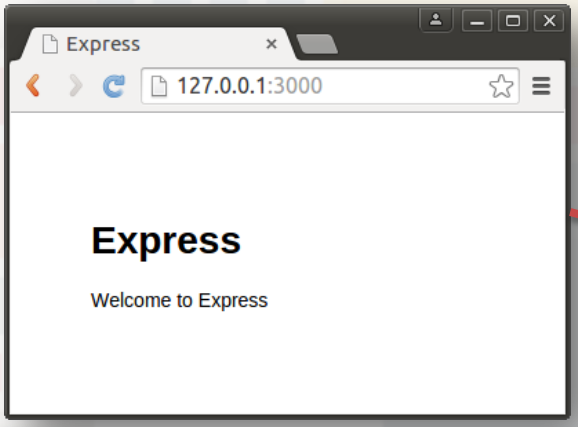
Step 6. Install dependencies

- Change the current directory to `myapp` with "**`cd myapp`**"
- Install dependencies with "**`npm install`**"

```
$ cd myapp/
$ npm install
cookie-parser@1.3.3 node_modules/cookie-parser
├── cookie@0.1.2
└── cookie-signature@1.0.5

debug@2.1.1 node_modules/debug
└── ms@0.6.2

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)

body-parser@1.10.2 node_modules/body-parser
├── media-typer@0.3.0
├── bytes@1.0.0
├── raw-body@1.3.2
├── on-finished@2.2.0 (ee-first@1.1.0)
├── depd@1.0.0
├── qs@2.3.3
├── iconv-lite@0.4.6
└── type-is@1.5.5 (mime-types@2.0.7)
...
```

Step 7. Run the app

- Let's run the app to see what has been created
- Execute "`DEBUG=myapp .\bin\www`"
- 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




```
$ DEBUG=myapp ./bin/www
GET / 200 386.303 ms - 170
GET /stylesheets/style.css 200 7.219 ms - 110
GET /favicon.ico 404 44.211 ms - 1116
```

Congratulations!

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

– End –