Setting up Node.js

Agenda

- Installing Node.js
  - Windows & macOS
  - Linux
  - Testing your Node.js and NPM installation
- Using NPM
  - package.json
  - Adding dependencies
  - Development dependencies
  - Running tasks
- Local Packages
- Global Packages
- Using NVM

Installing Node.js

- Windows & macOS
  - Download Node.js installer
    - Go to https://nodejs.org/en/
    - Select the button to download the LTS build (recommended)
    - Double-click on the downloaded file
    - Follow the installation prompts
- Linux (Ubuntu)
  - Install the most recent LTS version of Node.js using the package manager to get it from the Ubuntu binary distributions repository. Run these commands

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Installing Node.js

- Testing your Node.js and NPM installation
  - run the "version" command in your terminal/command prompt and check that a version string is returned

```
> node -v
```

  - NPM can also be tested in the same way

```
> npm -v
```

Using NPM

- NPM
  - Used to fetch packages that an app needs for development, testing and production
  - Also used to run tests and tools used in development process
- package.json
  - Text file used to manage dependencies for Node.js package / app
  - Includes package's name, version, description, initial file to execute, production dependencies, development dependencies, etc.
  - Contains everything NPM needs to fetch and run your app

Using NPM

- Adding dependencies
  - Create a directory for your new app and navigate into it

```
mkdir my-app
cd my-app
```

  - Use the ‘npm init’ command to create a package.json file for your app
    - Enter values for the various prompts
    - Press ‘Enter’ to accept default values
  - Observe that ‘package.json’ is created in ‘my-app’ directory. Check its contents

Using NPM

- Adding dependencies (Continued)
  - As an example, let us install the Moment JavaScript library in the my-app directory

```
npm install moment --save
```

    - --save option adds a ‘dependencies’ entry within package.json and includes a reference to ‘moment’ library

  - To use the library you call the require() function

```
const moment = require('moment');

const now = moment();
console.log(now.format('Do MMM YYYY')); // 21st Mar 2018
console.log(now.format('ddd, h A')); // Wed, 10 PM
```

    - Create a file named **index.js** within "my-app" app directory
    - Copy the contents given above

Using NPM

- Adding dependencies (Continued)
  - Run index.js using ‘node index.js’ command
    - Following output is displayed

```
21st Mar 2018
Wed, 10 PM
```

- Development dependencies
  - To add a dev dependency to our Node.js app, following command can be used

```
npm install http-server --save-dev
```

    - --save-dev option adds a ‘devDependencies’ entry within package.json and includes a reference to ‘http-server’ library

Using NPM

- Running tasks
  - You can define named scripts in package.json
  - Call NPM to execute named scripts
    - For e.g., to define a script to run the http-server development dependency, the following script block can be added to package.json

```
"scripts": {
  ...
  "serve": "http-server"
  ...
}
```

    - We would then be able to run http-server using NPM by calling:

```
npm run serve
```

Local Packages

- An app can consume a Node.js package to extend its functionality. It can reuse the functionality provided by the package if the package is installed locally
- This is ‘npm install’ command's default behavior

- Installing a local package
  - Use ‘npm install <package-name>’ command. See previous section (Adding Dependencies)
  - Creates node_modules directory in your app directory
  - Downloads the package to that directory

Local Packages

- Updating a local package
  - Sometimes we may need to update the local packages to get the improved code (new features, bug fixes, etc.)
  - Use ‘npm update <package-name>’ command
    - <package-name> is optional, if not specified, all the packages within package.json will get updated/upgraded to their newer versions
- Removing a local package
  - To remove a local package from node_modules directory, use ‘npm uninstall <package-name>’ command
    - Use ‘--save’ or ‘--save-dev’ option to remove the entry from package.json

Global Packages

- Certain Node.js packages are used as command line tools. Such packages should be installed globally
  - For e.g. Angular CLI, Grunt CLI, Http Server, webpack

- Installing a global package
  - Use ‘npm install -g <package-name>’ command
    - For e.g., to install ‘http-server’ package globally, execute this command

```
npm install -g http-server
```

    - When installed globally, the Node.js package works no matter which directory is current

Global Packages

- Updating a global package
  - Use ‘npm update -g <package-name>’ command
    - <package-name> is optional, if not specified, all the global packages will get updated/upgraded to their newer versions
- Removing a global package
  - To remove a global package, use this command

    npm uninstall -g <package-name>

Using NVM

- NVM
  - A command line utility that provides the ability to switch between different versions of Node.js
    - For e.g., if you want to test your Node.js app with latest version of Node.js without uninstalling the stable version, NVM can be used

- Installing NVM for Mac OS & Linux
  - To install or update nvm, you can use the install script using cURL or Wget

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh | bash
wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh | bash
```

  - For more information on installation, visit
    - https://github.com/creationix/nvm/blob/master/README.md#installation

Using NVM

- Installing NVM for Windows
  - Download the latest installer from
    - https://github.com/coreybutler/nvm-windows/releases
  - Run nvm-setup.exe and follow the steps in the Setup wizard
    - Note:
      - You need to uninstall any existing versions of node.js before installing NVM for Windows
      - Also delete any existing Node.js installation directories
  - For more information on installation, visit
    - https://github.com/coreybutler/nvm-windows#installation--upgrades

Using NVM

- Usage – NVM for Windows
  - To get help, use ‘nvm’ command
  - nvm arch [32|64]
    - Displays whether Node is running in 32 or 64 bit mode
    - Specify 32 or 64 to set the mode
    - Some examples

```
> nvm arch              // displays the mode
> nvm arch 32           // sets mode to 32-bit
> nvm arch 64           // sets mode to 64-bit
```

Using NVM

- Usage (continued)
  - nvm list
    - Lists the node.js installations
  - nvm list available
    - Shows the list of versions available for download
  - nvm install <version> [arch]
    - <version> can be a Node.js version or ‘latest’ for the latest stable version
    - [arch] is optional, it can be either 32 or 64

```
> nvm install 8.10.0 64   (installs Node.js version 8.10.0 64-bit)
> nvm install latest 64   (installs latest 64-bit version)
```

Using NVM

- Usage (continued)
  - nvm uninstall <version>
    - Uninstalls a specific version

```
> nvm uninstall 8.10.0  (uninstalls Node.js version 8.10.0)
```

  - nvm use <version> [arch]
    - Switch to use the specified version
    - [arch] is optional

```
> nvm use 8.10.0 32  (switches to Node.js version 8.10.0 64-bit)
```

  - nvm version
    - Displays the current running version of NVM for Windows

Using NVM

- Usage (continued)
  - nvm on
    - Enables node.js version management
  - nvm off
    - Disables node.js version management

THANK YOU