NPM

Node Package Manager

What is NPM

- When building large applications it’s important to split you project to multiple modules/packages
- with npm you can
  - Use community packages
  - Publish your package
  - Maintain package version and easily update the package you are using
- npm started as NodeJs backend package manager but today it’s also used to install frontend package

Install NPM

- Install NodeJs - npm arrives with node
- https://nodejs.org/en/
- you can verify npm is installed by typing: **npm -v**
- you can check node version by typing: **node -v**
- Angular requires node above v6 and npm above v3

Student EX.

- Install node
- make sure that node works by checking the node version
- make sure that npm works by checking the version
- make sure your node and npm versions are compatible with angular requirements

package.json

- each project or module that has use of npm will have a **package.json** file
- The file will contain information on the current project you are creating
- The file will be in **JSON** format
- Information includes:
    - package name
    - package version
    - dependencies
    - devDependencies
    - peerDependencies
    - author
    - git repo
- to create **package.json** file: **npm init** or **npm init --yes**

package.json - important fields

- **name** and **version** together will form an identifier for the package and has to be unique
- **bugs** will hold the url and email to send bug reports
- **main -** entry point for your package
- **dependencies -** list of packages that your package depends on and will be installed when you install your package
- **devDependencies -** are not needed to run the package only for preparing it
- **peerDependencies -** packages that your package assumes that are already installed, npm will issue a warning if those packages are not installed
- **engines -** it’s common to specify the node engine that is required for the package

Student Ex.

- It’s important to also be comfortable in publishing a package of your own so our goal is to create a package which prints hello world to the console
- create a folder for your package
- init npm in that folder
- make sure to specify in the package.json a name and version

npm registry

- npm registry is where the packages are saved
- when using npm to install a package or to publish a package, npm will refer to the registry, to see from which registry your npm is set you can type:
  - **npm get registry**
- by default when you install npm it is set to: **https://registry.npmjs.org**
- to set a new url for your registry:
  - **npm set registry <registry-url>**
- It’s important for companies to have there private npm registry

Student EX

- make sure that your registry is set to the company private registry

Install NPM package

- a package can be installed local or global
- if installed locally the package will be added to the **node_modules** folder where you **package.json** file is located
- local: **npm install <package-name> --save/--save-dev**
- global: **npm install -g <package-name>** (might require admin privileges)
- must of the packages we will install locally, global package are usually used to add programs to the command line
- the **--save/--save-dev** will determine where to save the package version in the **package.json**
- it’s recommended not to push **node_modules** to the repository
- by default

Uninstall Package

- npm uninstall <package-name> --save
- npm uninstall <package-name> --save-dev
- npm uninstall -g <package-name>

Scoped Packages

- namespaces for npm modules
- used for grouping related packages together
- scope begins with @
- recommend to prefix company private packages with prefix **@hcl**
- you can create a private npm repo and make all the scoped packages be pushed to private repo
- https://github.com/verdaccio/verdaccio

publishing your package

- After you finished creating your package you can try and publish it to the private npm registry
- to publish your package:
  - **npm publish**
- you might need to create a user to publish a package
  - **npm adduser**
  - or login if you have a user: **npm login**

Student Ex

- add implementation for your hello world package
- try to push it to the company npm private registry
- add a scoped package name with your name to the package
- make sure you have an account in the private registry
    - **npm adduser / npm login**
- Read about **nvm** and how to make the team work with the same node and npm version using **.nvmrc**
-

Summary - best practices

- in your angular projects, consider which which items will be required across multiple projects, those should be separated to different packages
- when creating a package specify in the **package.json** the **engines** for the node versions required
- make sure to use in your team **nvm** and **.nvmrc** to force the team to work with the same node and npm versions
- Usually a company decides of a scoped package of the company **@isracard/hello-world**
- you can also set up that packages scoped with: **@isracard** will be published to the private registry and all other packages will be published to the public registry