NPM - Node Package Manager

What is NPM

When you are creating your project, it's important to break your project into small modules/packages. A package is often small and is a building block that solves a certain problem that often recurs when you build your project, thus this building block can be used on multiple project or on multiple other packages you are building.
With NPM you can do the following:

- Use community packages
- Publish your packages
- Maintain versions of packages and easily update the packages you are using

NPM started as a NodeJS backend package manager but today it is also used for frontend packages.

Install NPM

When you install NodeJS you are also installing NPM.
For windows there is an easy windows installer in the **NodeJS** page.
For mac users it's recommended to use **Homebrew**

Student EX.

Install the latest version of **NodeJS**
Make sure after you install it that you can open the **CMD** and type:
npm -v
this command will show you the current version of node that is installed on your computer.

Package.json

each project or package you will create will contain a special file called **package.json**.
This file will contain information about the current package you are creating. Some of the important information this file includes is:

- package name
- package version
- dependencies - the packages that are required in order to use this package
- devDependencies - the packages that we need to install in order to contribute and develop this package

- Author
- Git repo

…

Let's create a package which contains a function which gets a string param and converts all the letters to capital letters.
We will start by creating the folder for our package somewhere:
mkdir capital
cd capital
npm init

This will automatically create a **package.json** file in the folder we created, by asking you a few question about your package. Each package you create has to contain a **package.json** file. Answer the questions that you are asked and your should see a **package.json** file created for you.

Student Ex.

We will create our first npm package. Create a folder for your package and a **package.json** file describing your package.

Install NPM Package

You can choose to install a package locally which will be available only in your project, or you can install a package globally that will be available in all projects in your computer.
To install a package locally, go to the folder where you created your **package.json** file and type in the terminal or CMD:

npm install lodash --save

When we are installing a package it's important that we put either the **--save** flag or **--save-dev** flag, this flag will modify the **package.json** to include the package we installed.
The package will be installed along with all the dependencies and it will be put in the **node_modules** folder.
It's recommended to place the **node_modules** folder in the **.gitignore** file.
When another developer clones the project he will not get the **node_modules** folder and will just get the **package.json** file. By typing in the directory where the **package.json** is located the command:
npm install
all the packages that are located in the **package.json** will be installed.

You can now type:

node

this will open the **nodejs** interpreter and you can type command and also use the packages you just installed:

var lodash = require('lodash');
…

Uninstalling a package

you can also uninstall a package by typing:

npm uninstall lodash --save

notice that it's important to add the **--save** and **--save-dev** so the changes will reflect in the **package.json** file.

Scoped Package

scopes are like namespaces for npm modules.
The scope begins with **@** for example the scope for **angular2** packages is: **@angular**
Each recommended that every private package you create in the company will have a scope.
For example **@migdal/web-styles**
Recommended setup for company is to create a private npm repo (either with your own server or with a paid subscription to **npm**) and set all the scoped packages that starts with **@migdal** to be pushed to the private repo and also if we want to install a **@migdal** package it will be taken from the private repo.
Every other package will be taken from the public community repo.

Student Ex.

- Create your package which contains a single function that gets a string and turns it to capital letters.
- notice that you have to name the file like the entry point file in the **package.json**
- notice that in order for you to use the function in the package you will have to attach it to the **exports** objects
- Change the name of the package from the **package.json** file to include a scope of **@migdal**

Creating your private NPM repo and pushing your package to it

We want to create a private npm repo and push packages with the scope **@migdal** to our private repo.
to create our npm repo we will use a package called **sinopia**
install **sinopia** by typing:

npm install -g sinopia

now you can run sinopia by typing:

sinopia

let's add a user to our local server by typing:

npm set registry http://localhost:4873
npm adduser --registry http://localhost:4873

Fill the questions to create a user for your private repo.
now you can publish your package to your private repo by typing:

npm publish --registry

you can now point your browser to **localhost:4873** and check out the package you just published

You can also set that all the scope packages will be published to the private repo by modifying the sinopia config. Edit the config by typing:

> nano ~/.config/sinopia/config.yaml

change the **packages:** section to this:

```
packages:
  '@migdal/*':
    # scoped packages
    access: $authenticated
    publish: $authenticated
```

and this will allow you to maintain your own private repo only for your scoped packages.

Summary

Developing a software today is not an easy task, especially when our software is getting more robust and more complex. NPM encourage us to try and split our big project to small building block called packages, which further encourage us to split our project to micro services.
I think it’s imperative for company to have some sort of solution to create their own npm private repository other as a paid service or just easily create your own byu using **sinopia** (we saw how easy it is).