- ***Back End Development and APIs***
  - -> using JS on the front-end -> for interactivity, building a SPA
  - -> using JS on the back-end -> the server or to build entire web applications
    - -> building applications through microservices
      - many modular applications that work together to create the entire application
  - ***-> this course***
    - -> writing back end apps with Node.js and npm
    - -> web applications with the Express framework
    - -> building a People Finder microservice with MongoDB and the Mongoose library
- ***Managing packages with NPM***
  - -> npm <- Node Package Manager
  - -> for installing, creating and sharing js packages
  - ***-> there are open source packages you can use***
    - -> check to see if someone has already written what you are creating
    - -> fetching data from an API
  - ***-> this course***
    - -> npm
    - -> how to work with package.json
    - -> how to manage installed dependencies
- ***How to Use package.json, the Core of Any Node.js Project or npm Package***
  - -> clone the github repo at https://www.freecodecamp.org/learn/back-end-development-and-apis/managing-packages-with-npm/how-to-use-package-json-the-core-of-any-node-js-project-or-npm-package
  - -> solve the challenges locally
  - -> use a site builder of your choice
  - -> the node.js file <- this is the js version of the index.html file
  - -> js is for the interaction of the user with the webpage, html is the content and css is the styling
  - -> it's json -> this has a certain syntax
  - -> the package.json file <- information is stored in key value pairs
    - -> name and version
    - -> the package.json file on the top level of the tree
    - -> the author field -> who created the project
      - -> this is an object which stores this information, but it can contain more simple information in the form of a string
      - -> "author": "Jane Doe", <- syntax
  - -> the task is to clone the code and to write your name in the package.json file, then submit the url to its repository on the page whose URL is above
  - -> https://gitpod.io/#https://github.com/franpanteli/APIs-Managing-Packages-with-NPM
- ***Add a Description to Your package.json***
  - -> in the package.json file, add in a description
  - -> "description": "A project that does something awesome", <- syntax
  - -> these are used in node.js projects
  - -> "" for field names
- ***Add keywords to the package.json***
  - -> keywords
  - -> for SEO
  - -> it's an array of strings
  - -> we're adding it into the package.json file
  - -> "keywords": ["descriptive", "related", "freecodecamp"],
  - -> the solution link we are giving it is the link to the window for the web application - not to the entire gitpod page we are editing it on

- ***Add a license to the package.json***
  - -> where we tell users what they are allowed to do with the project
  - -> MIT is open source
  - -> there are copyright laws for the code in countries
  - -> you can state what users can and can't do or it's plagiarism
- ***Add a version to the package.json***
  - -> add this to the package.json file
  - -> for the version of the current project
- ***Expand Your Project with External Packages from npm***
  - -> package managers
  - -> dependency management
  - -> npm installs of the packages we want -> we tell it this under dependancies
  - -> the external packages it is dependent on
- ***Manage npm Dependencies By Understanding Semantic Versioning***
  - -> versions follow semantic versioning
  - -> standard software versioning -> SemVer
  - -> this is useful when the software you are versioning has dependancies
  - -> "package": "MAJOR.MINOR.PATCH"
    - -> major <- making incompatible API changes
      - -> these will create breaks in the code that worked before, but the others won't
    - -> minor <- for functionality in a backwards-compatible manner
    - -> patches <- for backwards-compatible bug fixes
  - -> instructions
    - -> in the dependencies version
    - -> of the package.json
    - -> change the version of @freecodecamp/example to match MAJOR version 1, MINOR version 2 and PATCH version 13
    - -> in other words -> 1.2.13
    - -> under the version of the dependancy we just added in
- ***Use the Tilde-Character to Always Use the Latest Patch Version of a Dependency***
  - -> you need it to be able to use different versions of the package
  - -> e.g for bug fixes
  - -> "package": "~1.3.8" <- the tilde (~) allows it to use the patches if there are any, for that package dependency
    - -> the external package which our project is dependent on
- ***Use the Caret-Character to Use the Latest Minor Version of a Dependency***
  - -> to use the latest version of a minor dependency
  - -> ^ <- this allows it to use future versions / updates for that dependency
  - -> caret
  - -> our code is using external packages
  - -> this tells it to use the future version of the packages if there are updates which we haven't updated our package.json file with
- ***Remove a Package from Your Dependencies***
  - -> managing dependancies
  - -> including external dependancies
  - -> to remove a package from the dependancies, delete it
  - -> you need the right amount of commas after removing it
  - -> // "@freecodecamp/example": "^1.2.13",