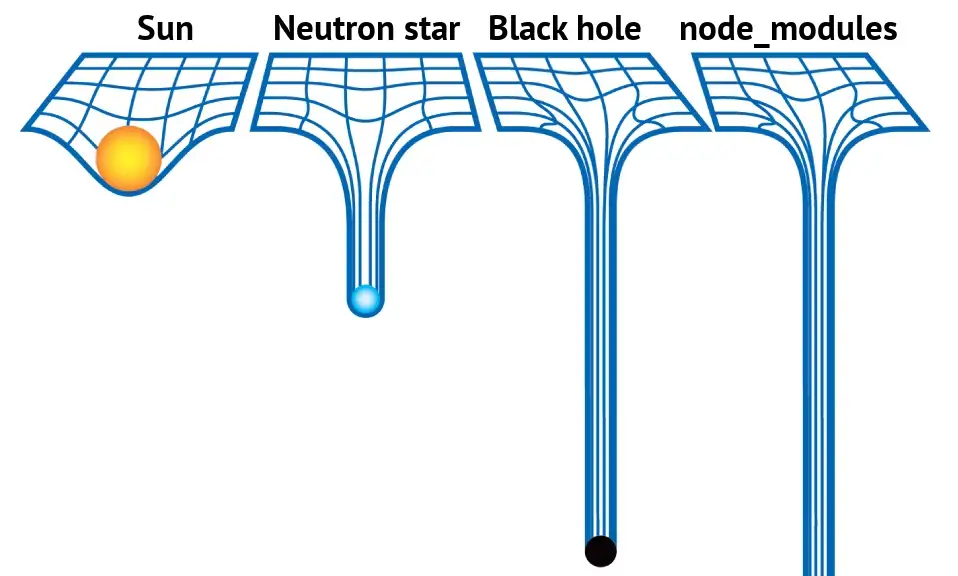
# Theory

* What is ‘NPM’?
  + npm is a package manager for the Javascript language. Originally, it stood for Node Package Manager but with the advent of JS and how far things have come, it does not (it is not necessary to use Node for npm).  
    It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.
* What is ‘Parcel/Webpack’? Why do we need it?
  + Parcel and Webpack are ‘bundlers’. You give them your files, including style files like Sass, Less or Stylus, your images, fonts, JavaScript files and they will assemble them in a seamless way, so they work perfectly on production. Both parcel and webpack allow the following features -
    - Minification
    - Image compression
    - HMR
    - Feature flagging (flag builds as dev, prod)
    - On top of this, we also get a development server, and diagnostics.
    - In addition to the above, Parcel also provides the following -
    - File watcher algorithm - made with C++
    - Cleaning our code (with the help of an additional package)
    - Super fast building algorithm
    - Caching while development
    - Compression of code (code obfuscation)
    - Old browser compatibility
    - HTTPS in dev
    - Different port numbers on builds on the same server but at the same time
    - Consistent hashing algorithm
    - Zero Configuration
    - Automatic code splitting
* What is ‘.parcel-cache’?
  + .parcel-cache is a folder used by ‘parcel’ to reduce build times. As the name suggests, it is a cache. It stores information about your project when parcel builds it, so that when it rebuilds, it doesn't have to re-analyze everything from scratch. It's a key reason why ‘parcel’ might be slower during initial building but is so much faster than other bundlers for subsequent builds.
* What is ‘npx’?
  + ‘npx’ is a tool that is used to execute the packages that have been installed using npm. Instead of installing a package system-wide, we can use ‘npx’ to execute the package (given that it is installed using npm in our node\_modules folder, of course)
* What is the difference between `dependencies` vs `devDependencies`?
  + "dependencies": Packages required by your application in production. These are needed by your application to run. Example - React, Express.
  + "devDependencies": Packages that are only needed for local development and testing. These are not needed by the application to run but might be required by developers to optimize the application. Example - Parcel, Mocha.
* What is Tree Shaking?
  + Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code (unused code). It relies on the import and export statements to detect if code modules are exported and imported for use between JavaScript files. If something is not used, it is removed from the optimized code.
* What is Hot Module Replacement?
  + When code is updated to include/exclude a module, this feature of a bundler makes sure that the application does not need a full reload.
* List down your favourite 5 superpowers of Parcel and describe any 3 of them in your own words.
  + Hot Module Replacement (HMR) - Update running build (dev) without a full application reload and allow for retention of state.
  + Tree shaking - Remove dead/unused code.
  + File watcher - Anytime code changes, the running server (dev) takes that into account and restarts.
  + Code compression (obfuscation) - Code files are compressed by removing unwanted spaces while still retaining what they do. The final code is not easy to understand but does exactly what it was meant to.
  + Caching - Rather than build everytime a build command is issued, ‘parcel’ creates a cache folder (.parcel-cache) and uses that for subsequent builds. This makes ‘parcel’ much faster than other bundlers.
* What is `.gitignore`? What should we add and not add into it?
  + ‘.gitignore’ is a plain text file in your repository that instructs git to ignore any files/folders that are added inside it. For example, if I wanted to instruct git to ignore the node\_modules folder, I will simply add node\_modules in .gitignore.
  + Files that do not need to be committed (passwords, keys) and files that can be generated by the server or are not needed by the server (node\_modules, .parcel-cache) should be added to it.
  + Files that are needed by the application should not be added. Example - package.json (to download packages needed for the application to run), code files, etc.
* What is the difference between `package.json` and `package-lock.json`
  + package.json:
    - This file is mandatory for every project
    - It contains basic information about the project (application name, author, repo, etc)
    - Contains the list of dependencies needed for the application to run
  + package-lock.json:
    - This file is automatically generated for those operations where npm modifies either the node\_module tree or package-json.
    - It is generated after an ‘npm install’
    - It allows future devs & automated systems to download the same dependencies as the project.
    - It records the version number of installed packages. Future installs will be capable of building identical description trees (no more ‘But it works on my machine’ since everyone is now capable of having the same dependencies and versions).
* Why should I not modify `package-lock.json`?
  + NPM changes the package-lock.json file itself. The file represents all the dependencies it has obtained since it may have received more up-to-date versions of some of them during npm install. Therefore, deleting or modifying the package-lock.json file in any way is akin to shooting oneself in the foot. Deleting it would cause dependency issues in the production environment.
* What is `node\_modules` ? Is it a good idea to push that on git?
  + All packages installed using npm (npm install) are installed not in your Applications folder but locally in the node\_modules folder.
  + It is not a good idea to push that to git.
    - It is possible that node\_module grows exponentially in size due to having tons of dependencies.
    - package.json and package-lock.json are the files being pushed to git because it will ensure that another execution of npm install will pull the required packages to another machine that is using the same repository.
* What is the `dist` folder?
  + The ‘dist’ folder contains the minimized version of the source code. The code present in the ‘dist’ folder is actually the code which is used on production web applications (thank you bundlers). Along with the minified code, it also comprises all the compiled modules that may or may not be used with other systems.
* What is `browserlists`?
  + The config to share target browsers and Node.js versions between different front-end tools. It is a simple variable in the package.json file or the .browserslistrc file and specifies that X, Y, Z are the browsers where the current web application should run. It is just a specification, not a compatibility layer.