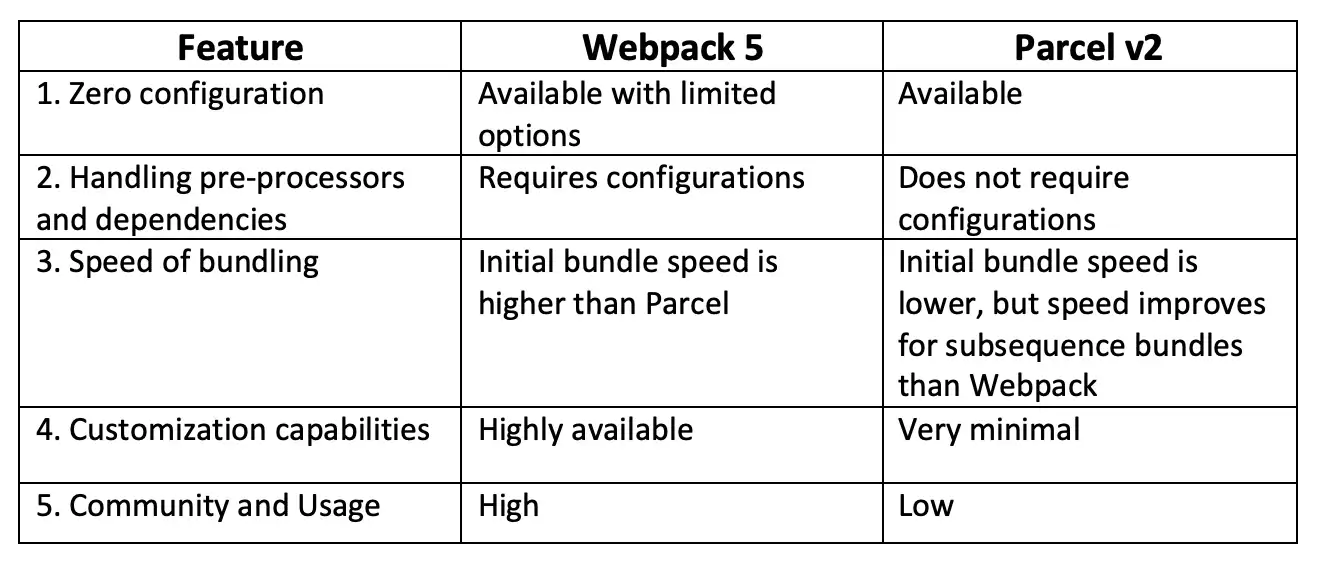
Assignment -2 (Igniting Our App)

1. What is NPM?

* NPM is the package manager for Node.js, there is no official abbreviation given for NPM.
* NPM is installed with Node.js, so you should have Node.js to get NPM installed.
* All npm packages are defined inside package.json in JSON format with version and addition fields.
* Npm can manage dependency and can be installed in the command line or terminal
* Dependencies are also defined inside the package.json
* If we want to share our own software with the registry you can sign in and do it

1. What is `Parcel/Webpack`? Why do we need it?

* Parcel/Webpack are bundlers mostly used for JS/Typescript code that helps you to minify, clean, and make your code compact so that it becomes easy to send request/response from the server.
* When you bundle your application initially, usually Parcel takes a considerable amount of time compared to WebPack. WebPack is faster. However, in subsequent builds (when you are watching and building), Parcel is much faster



1. What is a .parcel-cache?

* Parcel caches everything it builds to disk. If you restart the dev server, Parcel will only rebuild files that have changed since the last time it ran.
* Parcel automatically tracks all of the files, configuration, plugins, and dev dependencies that are involved in your build, and granularly invalidates the cache when something changes.
* For example, if you change a configuration file, all of the source files that rely on that configuration will be rebuilt.
* By default, the cache is stored in the .parcel-cache folder inside your project. You should add this folder to your .gitignore (or equivalent) so that it is not committed in your repo. You can also override the location of the cache using the --cache-dir CLI option.
* Caching can also be disabled using the --no-cache flag. Note that this only disables *reading* from the cache – a .parcel-cache folder will still be created.

1. what is npx?

NPX: The npx stands for Node Package Execute and it comes with the npm, when you installed npm above 5.2.0 version then automatically npx will installed. It is an npm package runner that can execute any package that you want from the npm registry without even installing that package.

1. What is difference between `dependencies` vs `devDependencies` ?

dependencies

Dependencies that your project needs to run, like a library that provides functions that you call from your code.

They are installed transitively (if A depends on B depends on C, npm install on A will install B and C).

*Example: lodash: your project calls some lodash functions.*

devDependencies

Dependencies you only need during development or releasing, like compilers that take your code and compile it into javascript, test frameworks or documentation generators.

They are not installed transitively (if A depends on B dev-depends on C, npm install on A will install B only).

*Example: grunt: your project uses grunt to build itself.*

Dependencies vs dev dependencies

Dev dependencies are modules which are only required during development whereas dependencies are required at runtime. If you are deploying your application, dependencies has to be installed, or else your app simply will not work. Libraries that you call from your code that enables the program to run can be considered as adependencies.

Eg- React , React - dom

Dev dependency modules need not be installed in the production server since you are not gonna develop in that machine .compilers that covert your code to javascript , test frameworks and document generators can be considered as dev-dependencies since they are only required during development .

Eg- ESLint , Babel , webpack

6. What is Tree Shaking?

Tree shaking is a term commonly used within a JavaScript context to describe dead code removal. It relies on the import and export statements to detect if code modules are exported and imported for use between JavaScript files.

7. What is Hot Module Replacement?

* HMR exchanges, add, or removes modules while an application is running, without a full reload, this can speed up development in a few ways.
  + Retain application state which is lost during a full reload
  + Save development time by updating only what changed
  + Instantly update the browser when modifications are made to CSS/js in the source code, which is almost comparable to changing styles directly in the browser's dev tool.

The following steps allow modules to be swapped in and out of an application:

1. The application asks the HMR runtime to check for updates.
2. The runtime asynchronously downloads the updates and notifies the application.
3. The application then asks the runtime to apply the updates.
4. The runtime synchronously applies the updates.

You can set up HMR so that this process happens automatically, or you can choose to require user interaction for updates to occur.

8. List down your favourite 5 superpowers of Parcel and describe any 3 of them in your

own words?

* Hot Module Replacement
* Bundling
* Tree shaking
* Minification
* Cleaning our code

9. What is `.gitignore`? What should we add and not add into it?

* Its text file that tells Git which files or folder to ignore in a project
* Local .gitignore file is usually placed in the root directory of a project
* We can also create global .gitignore file and any entries in that file will be ignored in all of your Git repositories
* To create local .gitingore file, create text file and name it .gitignore.
* The entries in this file can also follow matching pattern
  + \* is ised as a wildcard match
  + / is used to ignore pathnames relative to .gitignore file
  + # is used to add comments to a .gitignore file.
* To add or change your global .gitignore file
  + Git config –global core.excludesfile ~/.gitignore\_global
  + This will create file ~/.gitignore\_global

10. What is the difference between `package.json` and `package-lock.json` ?

* Package.json
  + Records only your direct dependencies and their versions
  + It contains information such as name, description, author, script, and dependencies.
  + It can be created using npm init or write into file with all information
  + It stores the minimum version required by your program. If you upgrade the versions of a certain package, the change will not be shown here.
* Package.lock.json
  + To know exact version we use in production we use package.log.json
  + If parcel version is upgraded in package.json then package.lock keeps snapshot of it
  + Very important file which locks version, you never have to keep it in git ignore
  + Maintain the exact version of it, keeps track of it maintain a hash of it.

11. Why should I not modify `package-lock.json`?

The reason is, package-lock. json may change automatically when you run npm install is because NPM is updating the package-lock. json file to accurately reflect all the dependencies it has downloaded since it may have gotten more up-to-date versions of some of them, Once NPM updates the package-lock.

12. What is `node\_modules`? Is it a good idea to push that on git ?

node\_modules is a directory created by npm and a way of tracking each packages you install locally via package. *Folder node\_modules should not be committed to Git. Apart from their big size, commits including them can become distracting and dependencies are system specific and might include native modules, you should never assume that your node\_modules folder will work in production.*

13. What is the `dist` folder?

* The shortform dist stands for distributable and refers to a directory where files will be stored that can be directly used by others without the need to compile or minify the source code that is being reused.
* Example: If I want to use the source code of a Java library that someone wrote, then you need to compile the sources first to make use of it. But if a library author puts the already compiled version into the repository, then you can just go ahead. Such an already compiled version is saved into the dist directory.
* Something similar applies to JavaScript modules. Usually JavaScript code is minified and obfuscated for use in production.
* Therefore, if you want to distribute a JavaScript library, it's advisable to put the plain (not minified) source code into an src (source) directory and the minified and obfuscated version into the dist (distributable) directoy, so others can grab the minified version right away without having to minify it themselves.

14. What is `browserlists`?

Browserslist is a tool that allows specifying which browsers should be supported in your frontend app by specifying "queries" in a config file

15. What difference between ^ - caret and ~ - tilda? Ans. The tilde matches the most recent minor version (the middle number) but caret will update you to the most recent major version (the first number).

~1.2.3 will match all 1.2.x versions, but it will miss 1.3.0.

^1.2.3 will match any 1.x.x release including 1.3.0, but it will hold off on 2.0.0.

16. What are the different types in the script?

* type="module" -> This value causes the code to be treated as a JavaScript module. The processing of the script contents is deferred. The charset and defer attributes have no effect.
* type="importmap" -> This value indicates that the body of the element contains an import map. The import map is a JSON object that developers can use to control how the browser resolves module specifiers when importing JavaScript modules.
* type="blocking" -> This attribute explicitly indicates that certain operations should be blocked on the fetching of the script. The operations that are to be blocked must be a space-separated list of blocking attributes.

**Parcel:**

Parcel Does below task:

HMR - Hot Module Releacing

File Watcher elgoritta C++

BUNDLING

MINIFY

CLeaning our Code

Dev and Production Build

Super Fast build algoritha

Image Optinization

A Caching while development

A Conpression

A Corpatole with older version of browser

HTTPS on dev

port Nurber

Consistent Hashing Algorithn

Zero Conrig