**Nodejs:**

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

link: https://nodejs.org/en/

**why we use gulp?**

Gulp is a task/build runner for development. It allows you to do a lot of stuff within your development workflow. You can compile sass files, uglify and compress js files and much more. The kicker for gulp is that its a streaming build system which doesn't write temp files. It's like Pipes in bash. One task flow into another. Gulp could be compared to Grunt in its usage.

link: http://brandonclapp.com/what-is-gulp-js-and-why-use-it/

**Expressjs:**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

link: http://expressjs.com/

**EJS:**

"E" is for "effective." EJS is a simple templating language that lets you generate HTML markup with plain JavaScript.

No religiousness about how to organize things.

No reinvention of iteration and control-flow. It's just plain JavaScript.

**Difference b/w dependencies and devdependencies:**

devDependencies will get installed from a npm install foo command.

These may include testing tools like mocha which are nice to have but not strictly needed to run the npm package at hand.

dependencies are must haves.

**Funtionality of following modules:**

1. gulp-rimraf:

It is used for delete the files.

2. gulp-typescript:

It is used for transpile ts files into js files.

3. gulp-nodemon:

It is used for auto restart the server.

4. gulp-watch:

It is identify if any changes in the code, or if any error in the code it will identify

**Middlewares:**

Express is a routing and middleware web framework with minimal functionality of its own: An Express application is essentially a series of middleware calls.

Middleware is a function with access to the request object (req), the response object (res), and the next middleware in the application’s request-response cycle, commonly denoted by a variable named next.

**Middleware can:**

Execute any code.

Make changes to the request and the response objects.

End the request-response cycle.

Call the next middleware in the stack.

If the current middleware does not end the request-response cycle, it must call next() to pass control to the next middleware, otherwise the request will be left hanging.

**Application-level middleware:**

Bind application-level middleware to an instance of the app object with app.use() and app.METHOD(), where METHOD is is the HTTP method of the request that it handles, such as GET, PUT, POST, and so on, in lowercase.