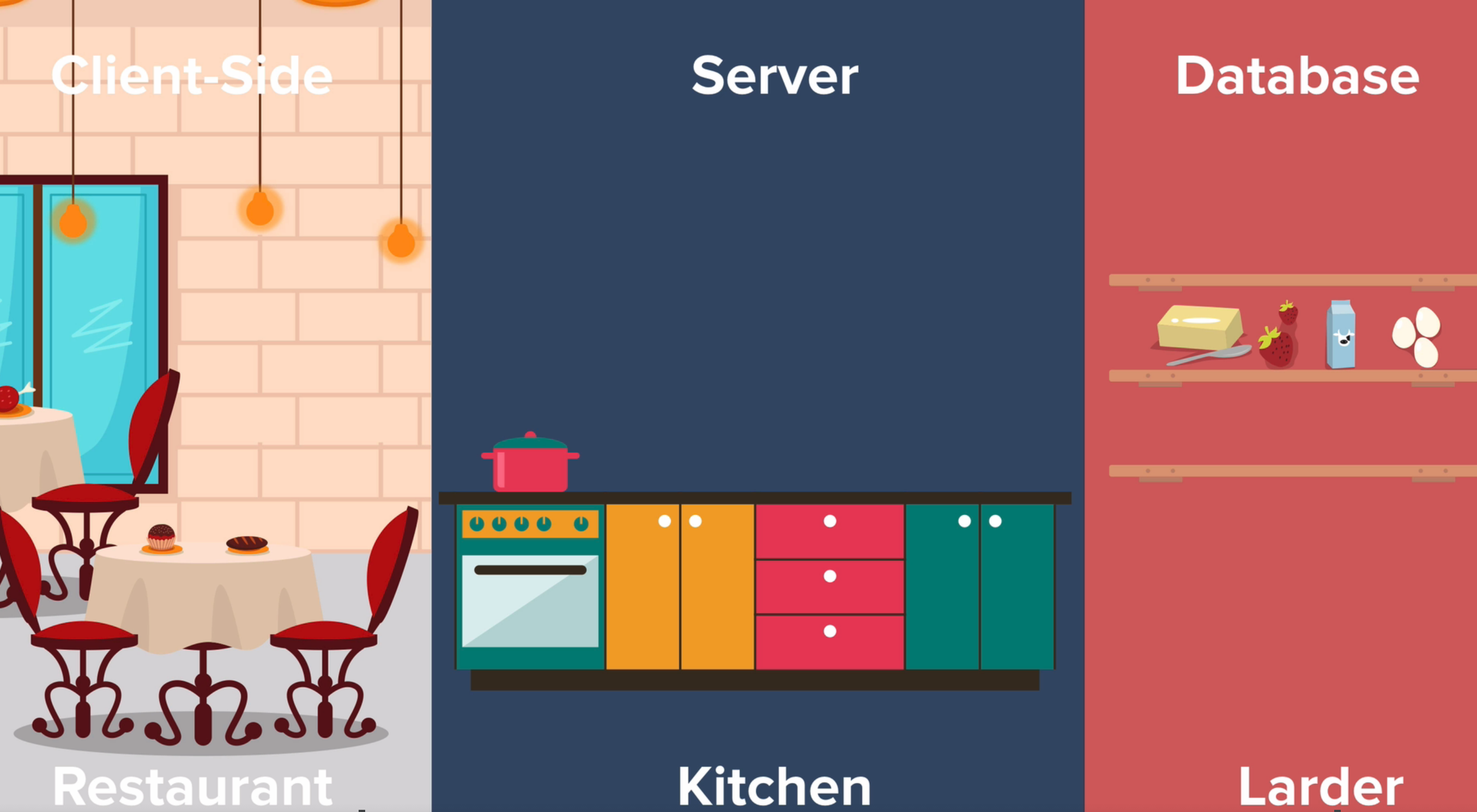
**Node JS**





Backend Technologies: Java, PHP, Ruby, Node.js

Frameworks: Spring, Express, Ruby on Rails, cake PHP, etc.

Node is super-fast and thus, allows us to create scalable and fast website.

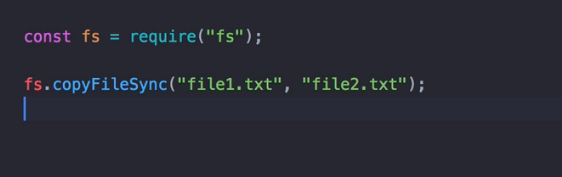
e.g., Uber, Netflix, etc.

Node REPL

* Node REPL allows us to run our program line by line as it was there in chrome developer tool.
* It stands for “ Read Evaluation Print Loops”
* It is invoked by simply typing node and pressing ‘Enter’ in Hyper.
* To exit, type ‘.exit’ or press Ctrl+C twice.
* We can use a number of internal node modules (native node modules bundled with node.js) to interact with our system directly using node. These node modules are present at <https://nodejs.org/api/>
* For example, we can access our file system using the node ‘File System’ module.

Note:

* Platforms to deploy your backend for free: Heroku, Firebase, AWS Free Tier, Google Cloud Free Tier, Glitch.

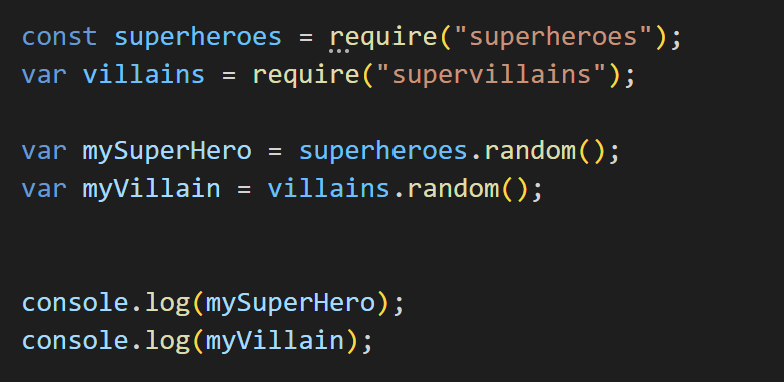


NPM

* NPM stands for “Node Package Manager”.
* NPM is a package manager for external modules.
* Packages in npm are bits of code written by somebody else to perform some specific functionalities which we can incorporate in our own projects.
* NPM comes bundled with node when we install node.
* NPM can be started with ‘npm init’.
* .json is a file format that organizes data.

Process of Using NPM

1. Use git init to initialize npm. It creates a package.json file.
2. Head over to npm website and download an external module or package. This gets added as a dependency in our project.
3. Call a require method to the module and store it in a variable or const.
4. Make the appropriate function call on the variable and you will get a response.



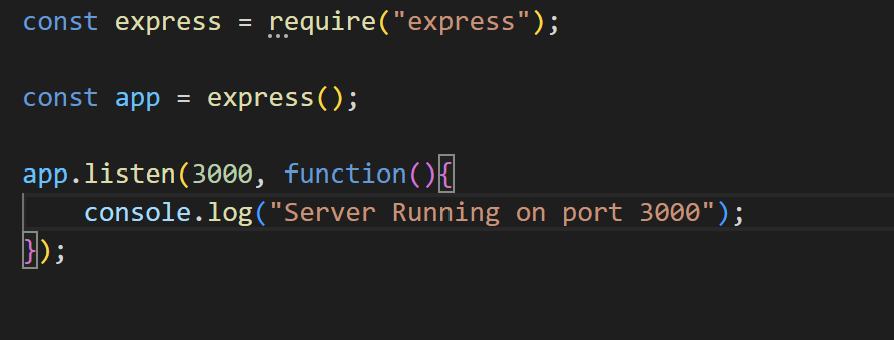
**Express**

* Express is a popular, minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is used for building APIs, web applications, and backend services.
* With Express, we can easily define routes for your application and handle HTTP requests such as GET, POST, PUT, DELETE, etc. It also supports middleware, which allows you to execute some code before or after a request is processed. This makes it easy to add functionality such as authentication, validation, and error handling.
* Express provides a simple way to get a web application up and running quickly, and it is widely used in the Node.js community due to its simplicity, robustness, and the number of available packages and plugins.

**Creating a server**

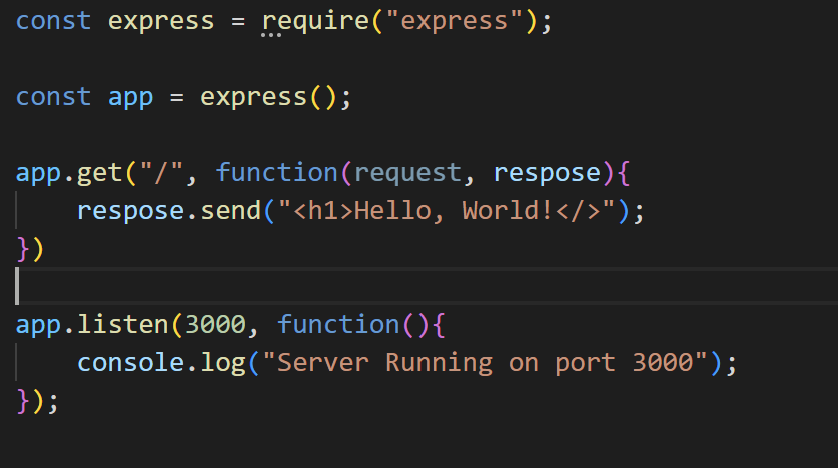


We can also add a call-back function to notify when the server is running.



**Handling Requests and Responses**

* We can run our server in our browser by specifying its location (here, localhost:3000).
* When the browser gets connected to the server, it sends a GET request to the server and the server send a response to it.



* The GET request has two inputs, first the location where the response it to be displayed (route) and a call-back function having two arguments request and response.
* The server sends a response using **‘send’** method.
* Here, ‘/’ displays home page or the root of website.

Note:

* To auto restart our server whenever there is a change in our source code, we can use ‘nodemon’ package.
* This can be installed using the command:

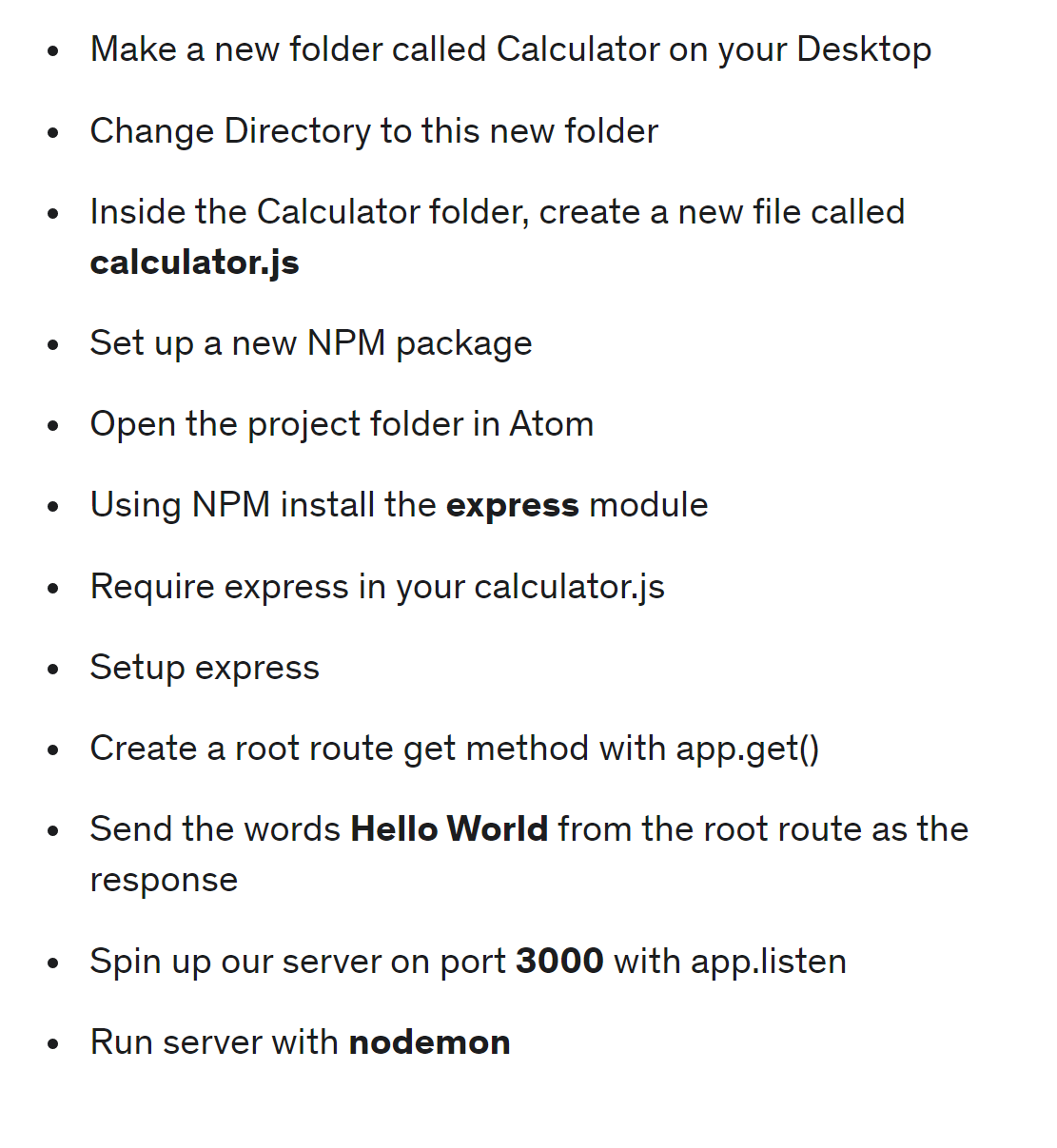
npm install -g nodemon

* And we can activate nodemon by getting into our server directory and using the command:

nodemon <serverName>

e.g., nodemon server.js

**Calculator Web App**



Note:

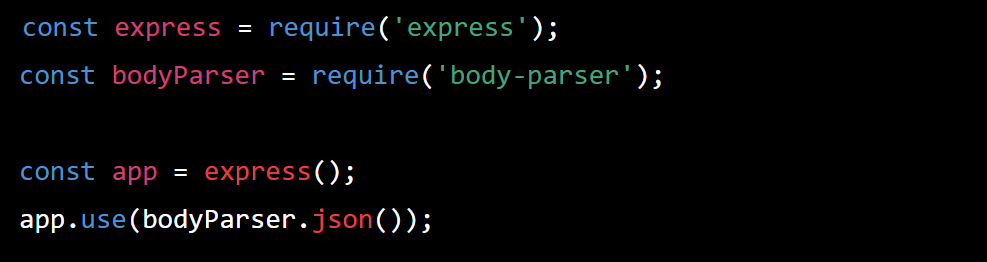
* To get the location of any given file, we can use a constant **\_\_dirname.**
* It is a constant which gives the name of the directory/ folder of the given file.
* Instead of using send method, we can use **sendFile(FileLocation)** to send a file.

Processing a POST request using body parser

The body-parser middleware in Express is used to parse the incoming request bodies and make it available as a property of the request object in your routes. It allows you to access data sent from a client (e.g., in an HTTP POST request) in your route handlers.

There are two main types of body-parsers in Express:

1. json: This parser can be used to parse JSON request bodies. To use this parser, you first need to install the body-parser package and then include it in your Express application as follows:



ii. urlencoded: This parser can be used to parse URL-encoded request bodies.

The information is sent to the browser using the post method. The POST method has two inputs, the location at which the data is to be sent and a call-back function having two arguments: request and response.

