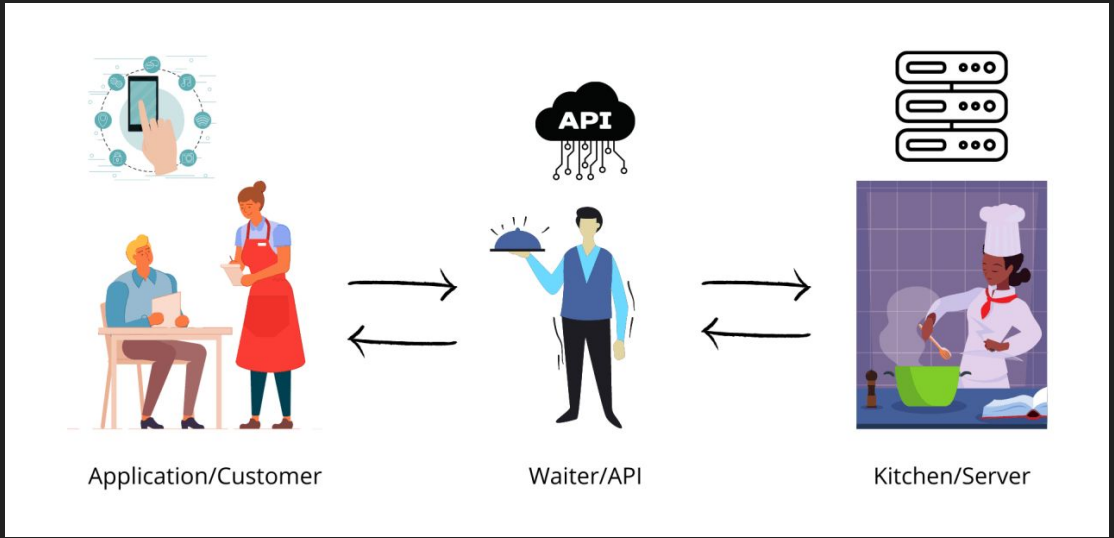
Welcome Geeks

Topics to be covered

- APIs
- Introduction to REST API
- Understanding CRUD & HTTP Verbs
- Introduction to Node JS
- Basic Hello World REST API
- Brief about Express JS

API

- Application Programming Interface



REST

Client: the client is the person or software who uses the API. It can be a developer, for example you, as a developer, can use Twitter API to read and write data from Twitter

Resource: a resource can be any object the API can provide information about. In Instagram's API, for example, a resource can be a user, a photo, a hashtag. Each resource has a unique identifier. The identifier can be a name or a number.

REST: architectural style

A RESTful web application exposes information about itself in the form of information about its resources.

It also enables the client to take actions on those resources, such as create new resources (i.e. create a new user) or change existing resources (i.e. edit a post).

REST

REST stands for **REpresentational State Transfer**.

It means when a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource.

For example, when a developer calls Instagram API to fetch a specific user (the resource), the API will return the state of that user, including their name, the number of posts that user posted on Instagram so far, how many followers they have, and more.

HTTP Verbs

The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently.

CRUD

HTTP Verb	CRUD
POST	Create
GET	Read
PUT	Update/Replace
DELETE	Delete

POST: The POST verb is most-often utilized to **create** new resources

GET: The HTTP GET method is used to **read** (or retrieve) a representation of a resource.
In the “happy” (or non-error) path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

CRUD

HTTP Verb	CRUD
POST	Create
GET	Read
PUT	Update/Replace
DELETE	Delete

PUT is most-often utilized for **update** capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

DELETE is pretty easy to understand. It is used to **delete** a resource identified by a URI.

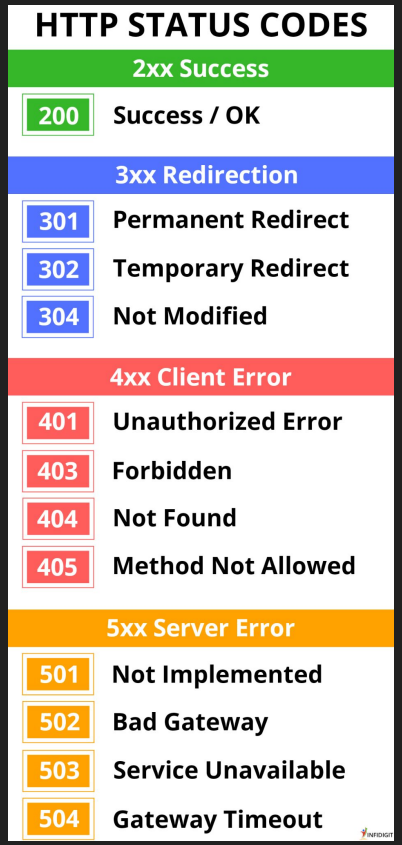
CRUD

```
fetch('https://jsonplaceholder.typicode.com/todos/1')
  .then(response => response.json())
  .then(json => console.log(json))

fetch('https://jsonplaceholder.typicode.com/posts/3', {
  method: 'DELETE',
});
```

When making a request using the fetch API, the default method is the GET method – so we are not forced to specify it. But we do need to specify when making a request using the other methods.

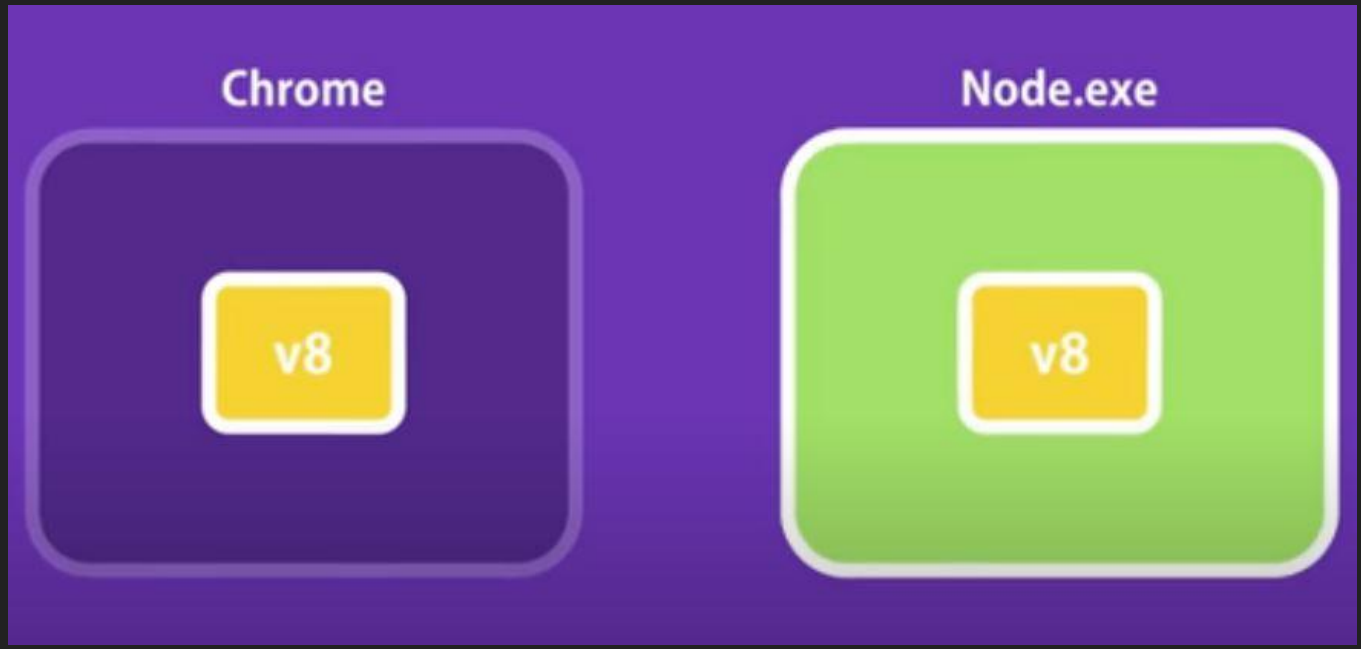
HTTP Status Codes



Introduction to Node.js

- Environment to run JS outside browser
- Built on Chrome’s V8 engine
- Not a programming language
- JavaScript everywhere

Introduction to Node.js



Introduction to Node.js

Browser

- DOM
- Window
- No file system

Node

- No DOM
- No Window
- File System

Installation of Node.js

HTTP MODULE

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper-Text Transfer Protocol (HTTP).

HTTP MODULE

Http Server

The HTTP module can create a client/server connection using http protocol.
We can create a new server using the method
http.createServer();
This creates an object of http.Server class and with this object we can use two methods:
• close() stops the server from accepting new connections
• listen() starts the HTTP server and listens for connections
The method createServer() takes a callback function with two arguments - request and response.

ExpressJS

Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js