# Section 20 → APIs → Application Programming Interfaces.

① You have seen weather reports on websites → They put APIs of National weather forcast Agency on their website. OpenWeather.org

- Also seen when you use social media and see mutual friends or interests.

② → Def → An API is a set of commands, protocols and objects a programmer can use to create software (or) interact with ext environment.
(like JQuery) (Like egs in ①).

- Simply said Websites/Company can let others use some data of their & keep some of it as private data.
- We can use some of that data on our website.

③ → APIs → Endpoints. → Paths. → Parameters. → Authentication. } Main Characteristics.

→ Endpoints

→ URLs to use that website; May have many.

→ Paths & Params → Specific paths (or) parameters (after specified the endpoint).

Paths → specified pages ( /bio , /about ).

Parameters → not specified custom pages.
(identified by ? contain = query (typed by user) can be anything)

④ Authentication → When we come to monitizable data, we have to seperate shareable & paid info.

→ Thats why authentication used.

Eg:- https://www.udemy.com/course/Questions?weather=what.

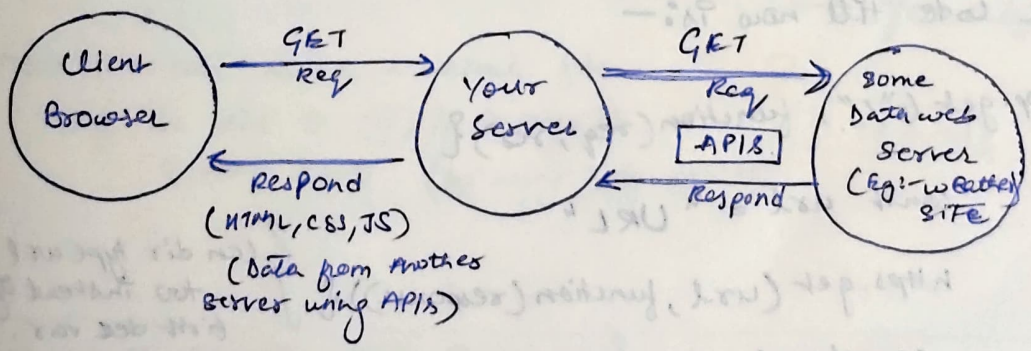
Endpoint. path: parameters

⑤ Postman (software) (optional) → to build API parameters by they contain many things.

⑥ JSON (JS object Notation).

- format in which we get data back from other websites.
- Very similar to JS (JavaScp) format.
- Can be other too i.e All are HTML, XML.
- Chrome web store → JSON (Viewer) Web Awesome. (optional)

⑦




→ const https = require ("https");

down → inside app.get

https.get ("website address", function() {

}) ;

can also add

8. → Parse JSON.

x → changeable

→ You can see https nodejs documentation for more.

→ response.on("data", function(data){
const a = JSON.parse(data)
console.log(a).
}

Inside https. converts data into JSON-format.

→ Opposite of parse is stringify.

9. How to pull specific data to display on our website.

const temp = a.main.temp

↳ Address/path of temp.

→ Can use JSON Viewer Awesome → copy path for this. (comes while you hover).

10. Now we have the data, lets put it on website.

→ So the code till now is:—

app.get("/", function(req, res){
const url = "URL"
https.get(url, function(response){
console.log(response.statusCode);
Now in response.on("data", ---)
}

Can directly type url here instead of first declared var.

for 401, 200, etc codes.

After this code is given in notes above.

(IV)

» go, »

To send it to website we know we use res.send.

* There can be only 1 res.send in app.get method.

» res.send("Temp Ps" + temp + "degrees.");

↳ can add html also.

» For multiple

```
res.write("....")
res.write("....")
res.send()
```

} will send multiple ....

* Inside (response.on) below (const temp)

» const icon = a.weather[0].icon
const imageURL = "https...."

} For adding image icon related to weather.

» (can see whole code at end of video if any confusion)

(V) Taking user input instead of our own.

» const city = "Paris"
const unit = "metric"
:etc

} we delete these things in URL and instead add these variables.

? contains = London

↳ ?contains=" + city + "

Now we can make a html file like we did in (3) in Section 19, & take inputs. (by using body-parser)

} 8:28 min.

---

» app.use(express.static("public"));

↳ can be any name

↳ for using static files from our comp. to be used on server.

18:00 – 20:00

Pn 248.