DAY 26

* Async Function:

async & await keywords

create async function :- (it return Promise object)

① async function greet()
{
    return "hello";
}

$$\begin{tikzcd} & F^n \arrow[dl] \arrow[dr] & \\ \text{Normal} \arrow[d] & & \text{error} \arrow[d] \\ \text{P: Fulfil + return} & & \text{P: rejected} \end{tikzcd}$$

② let greet = async () => {

}

* await keyword:

Pause the execution of its surrounding async function until the the promise is settled (resolved or rejected)

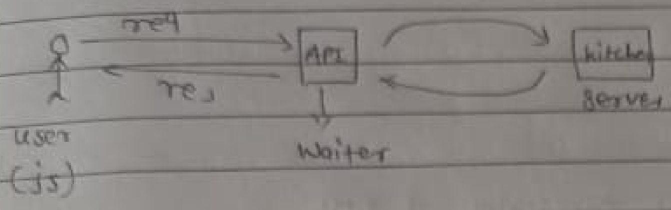
e.g async function show()
{
    await Colorchange ("violet" ~~"voilet"~~, 1000);
    await colorchange (" lime", 2000);
    return "done";
}

* Handling Rejection with Await

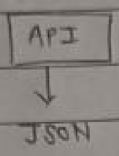
* use try_Catch block

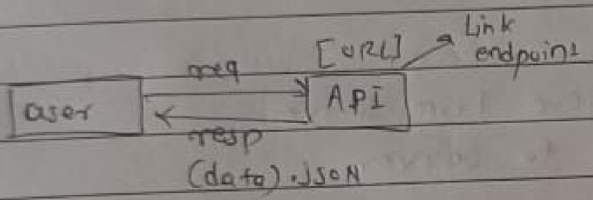
```
let h1 = document.querySelector("h1");
Function changeColor(color, delay)
{
    return new promise((resolve, reject)=>{
      SetTimeout(() => {
          let num = Math.Floor(Math.random() * 5) +1;
          if(num >3)
          {
              reject("Promise reject");
          }
          h1.style.color = color;
          Console.log(color changed to ${color}`);
          resolve("Color Changed");
      }, delay});
    });
}
async function color() {
 try{
  await("Violet", 10000);
  await("Pink", 20000);
  await("lime", 1000);
  await("Purple", 1000);
 }
Catch(err)
{
 Console.log("error Caught');
 Console.log("err);
}
 Console.log(2);
}
```

* API :- Application Programming Interface



API return data in JSON Format





* What is JSON :

Javascript object notation www.json.org

· e.g

```
{
    "fname" : "Rohit",
    "lname" : "Katkar"
}
```

* Accessing Data form JSON :

json data → js object

1) JSON.parse(data) Method

To parse a string data into a js object.

js Object → json data

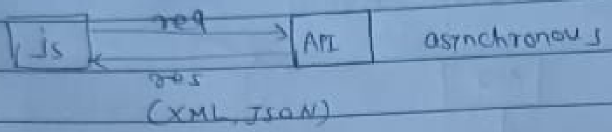
2) JSON.Stringify (json) Method

To Parse a js object data into JSON.

* Testing API request :-

✓ 1) Hoppscoth
2) Postman

* Ajax : Asynchronous javascript & XML



* HTTP verbs :

· Get	Retrieve From Server
· POST	Send to Server
· DELETE	Delete Something

* Status Code :

200	– ok
404	– Not Found
400	– Bad request
500	– Internal server error

Informational response (100 - 199)
Successful response (200 - 299)
Redirection message (300 - 399)
Client error response (400 - 499)
Server Error response (500 - 599)

* Add information in URL.

Query string:

https://www.google.com/search?q=harry+porter

key value

?name=Rohit&marks=95

* http ~~header~~ header

header, value

Accept	:	text/Plain/html
		appsication/JSON

* Our First Request

Using Fetch

```
let url = "https://catfact.ninja/fact";
Fetch(url);          // return promise

res.json()           // retur data in readable format
```

Async Functions

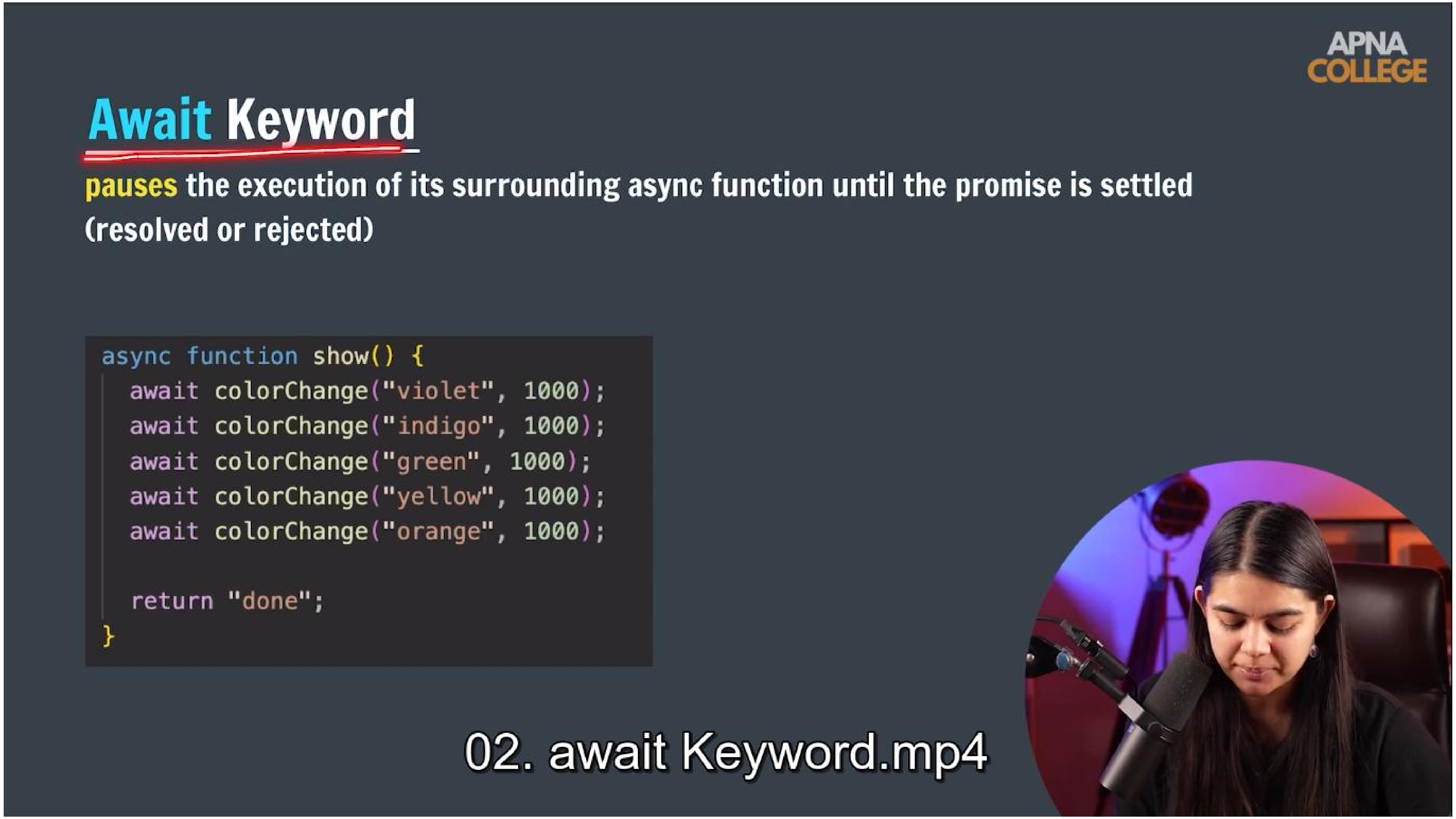
async & await Keywords

01. async Functions.mp4

Async Keyword

Creates an Async Function

```
async function greet() {
  return "hello world!"; //returns a promise
}

let hello = async () => {}; //returns a promise
```



Await Keyword

pauses the execution of its surrounding async function until the promise is settled (resolved or rejected)

```
async function show() {
  await colorChange("violet", 1000);
  await colorChange("indigo", 1000);
  await colorChange("green", 1000);
  await colorChange("yellow", 1000);
  await colorChange("orange", 1000);

  return "done";
}
```

02. await Keyword.mp4

Await Keyword

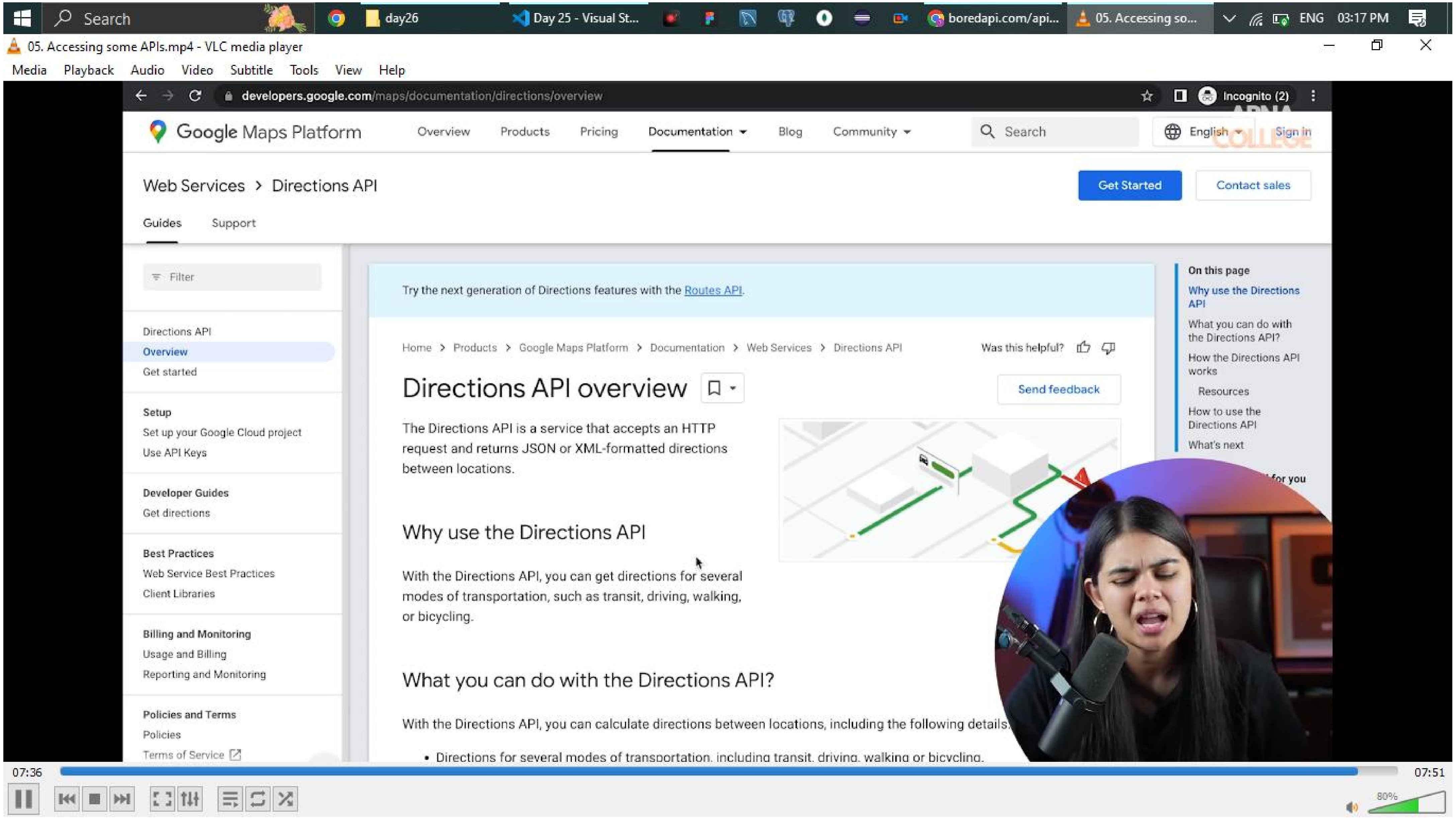
Handling Rejections with Await

$\Longrightarrow$

API

Application Programming Interface

JS


05. Accessing some APIs.mp4 - VLC media player

developers.google.com/maps/documentation/directions/overview

Google Maps Platform

Web Services > Directions API

Try the next generation of Directions features with the Routes API.

Home > Products > Google Maps Platform > Documentation > Web Services > Directions API

Directions API overview

The Directions API is a service that accepts an HTTP request and returns JSON or XML-formatted directions between locations.

Why use the Directions API

With the Directions API, you can get directions for several modes of transportation, such as transit, driving, walking, or bicycling.

What you can do with the Directions API?

With the Directions API, you can calculate directions between locations, including the following details:

- Directions for several modes of transportation, including transit, driving, walking or bicycling.

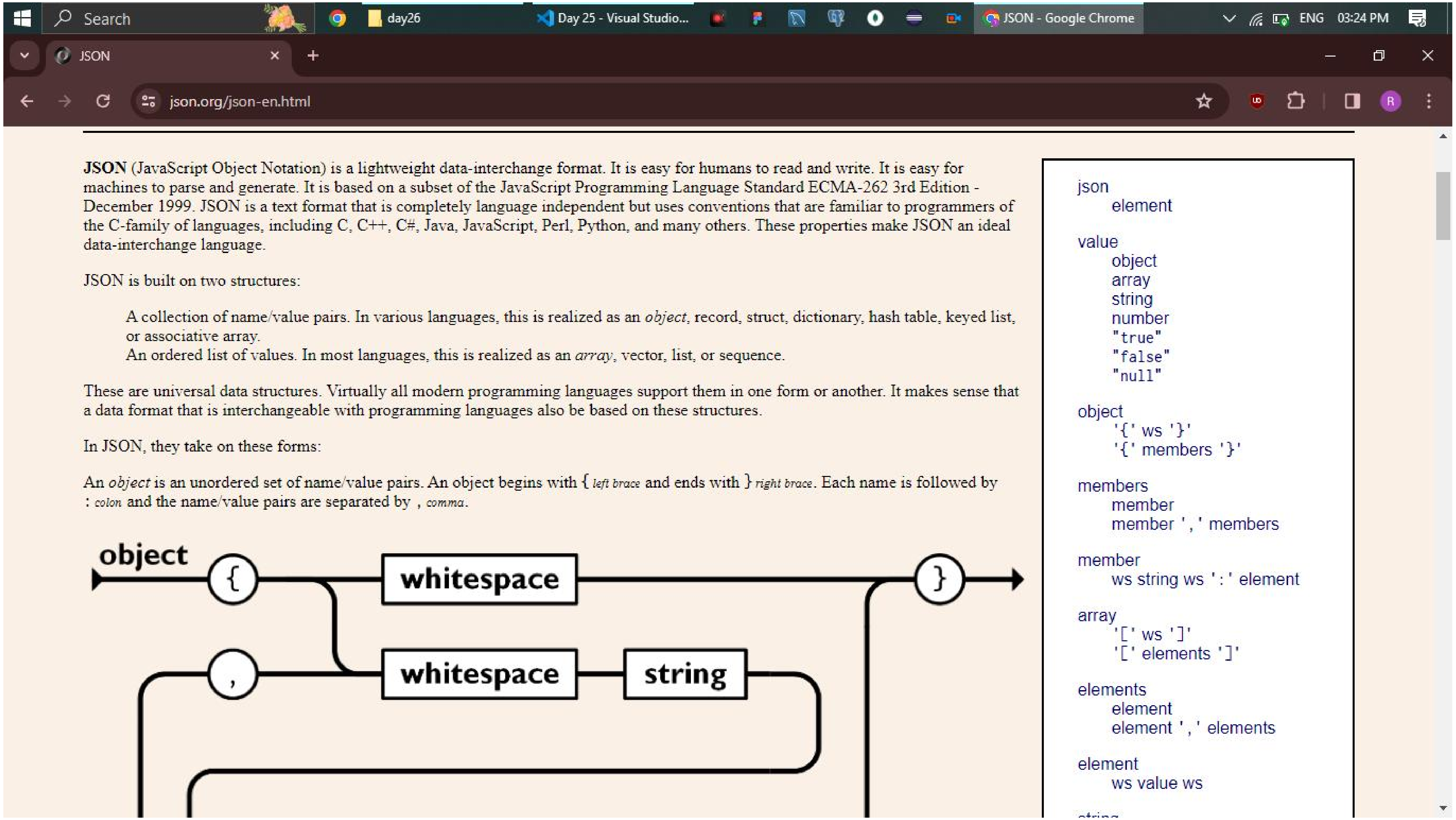
APNA COLLEGE

JSON

JavaScript Object Notation   *www.json.org*

06. What is JSON_.mp4

JS



JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

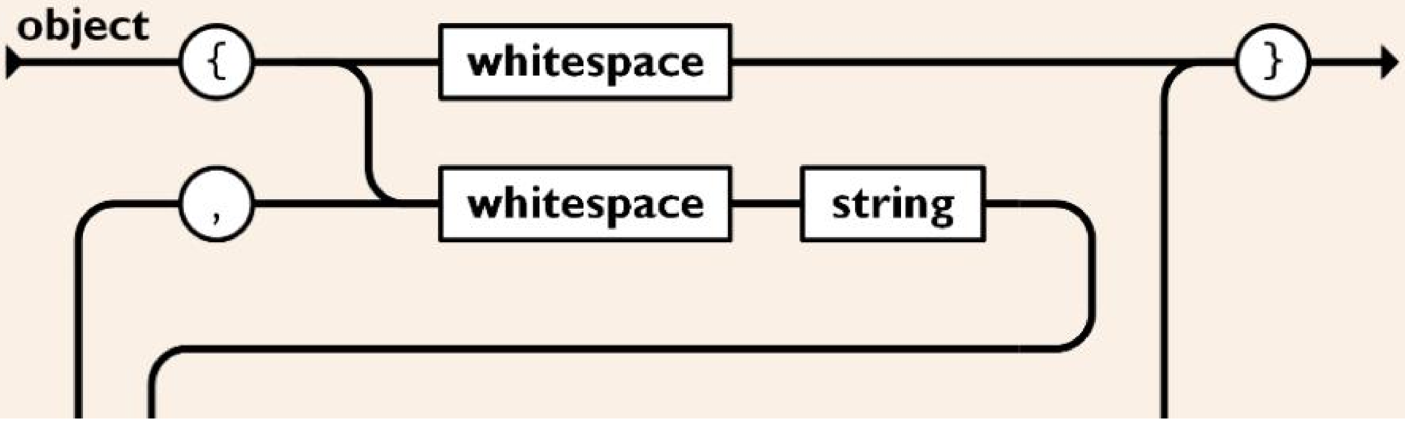
JSON is built on two structures:

> A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
> An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

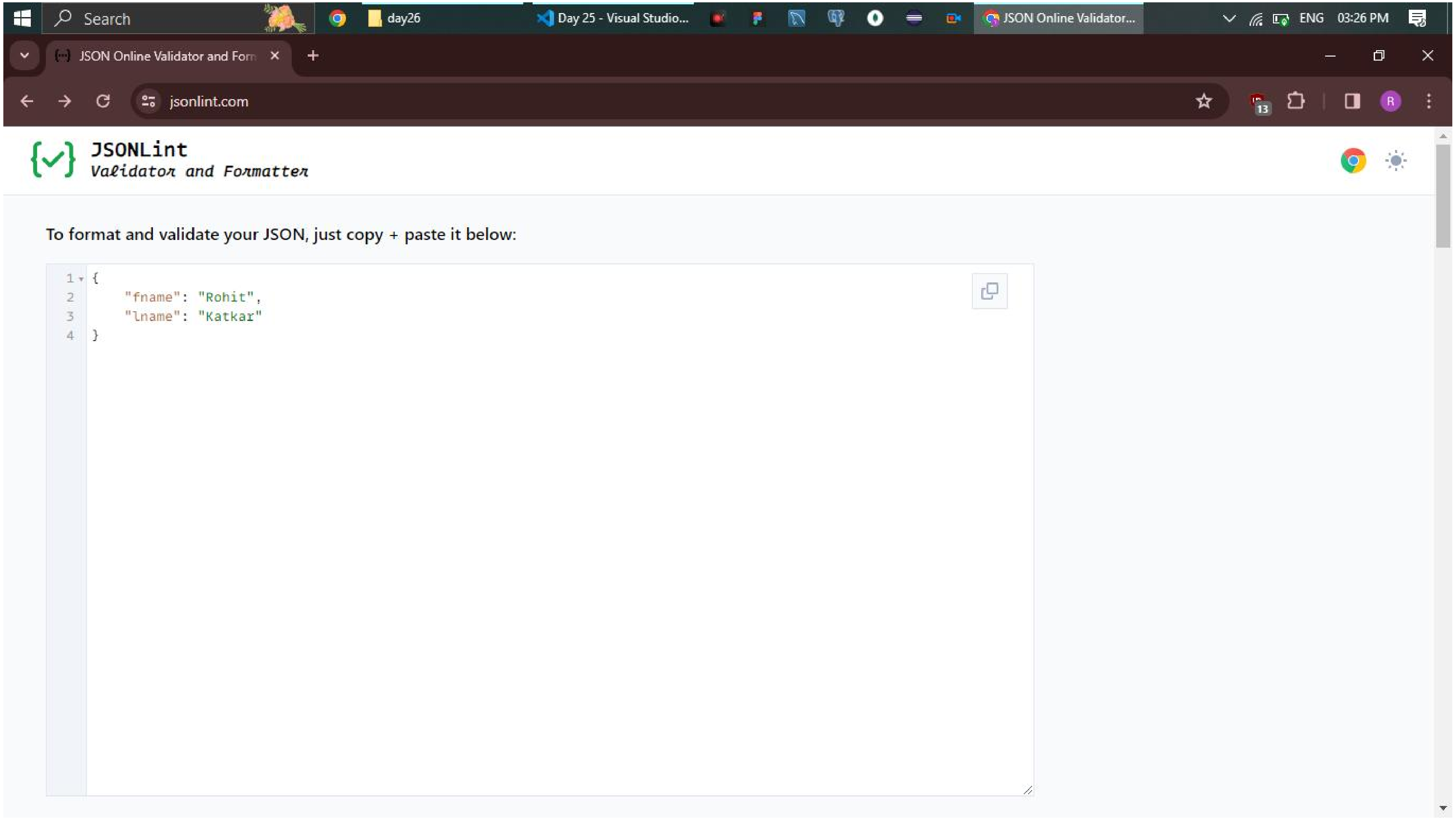
These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { *left brace* and ends with } *right brace*. Each name is followed by : *colon* and the name/value pairs are separated by , *comma*.

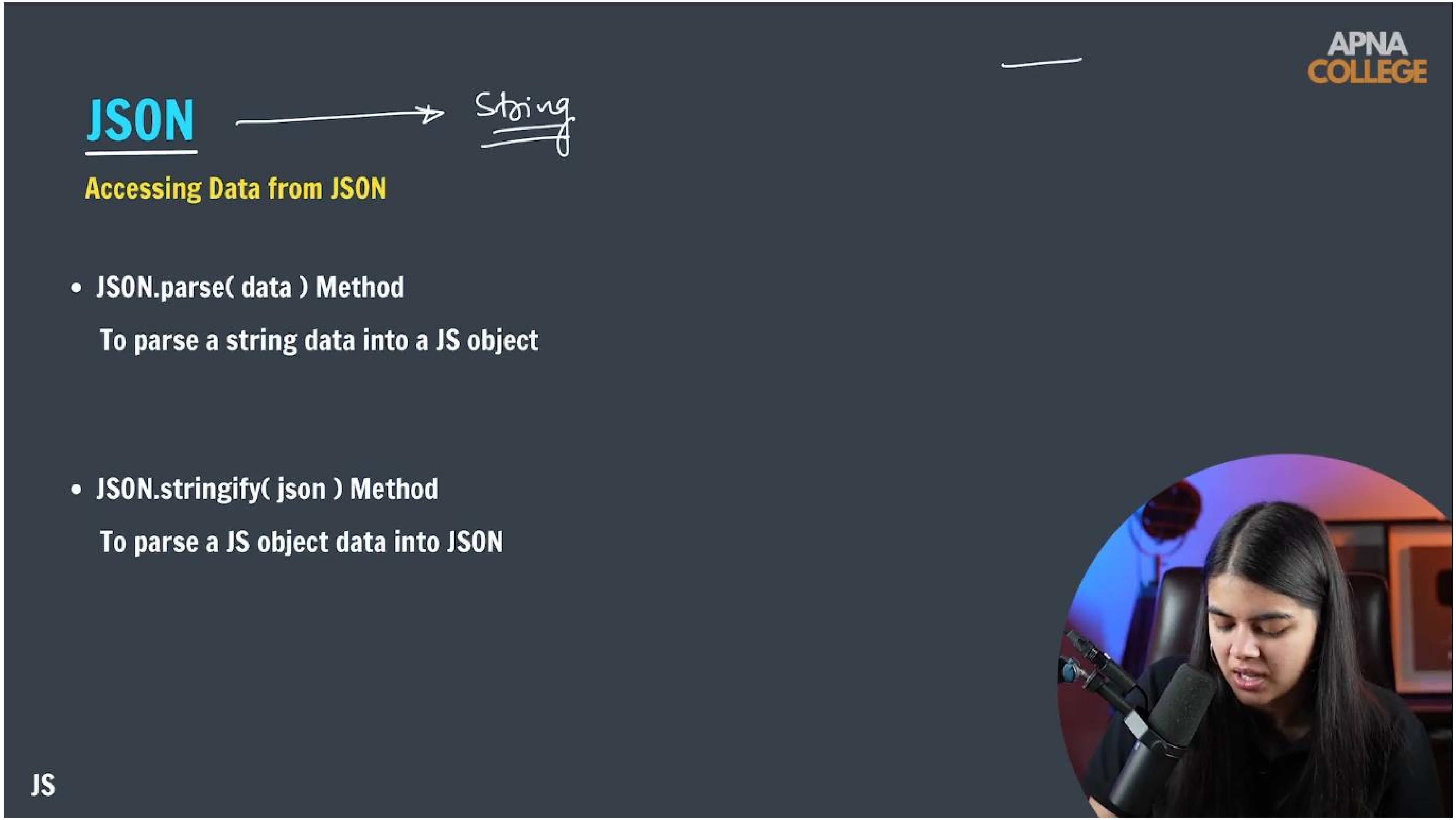


```
json
    element

value
    object
    array
    string
    number
    "true"
    "false"
    "null"

object
    '{' ws '}'
    '{' members '}'

members
    member
    member ',' members

member
    ws string ws ':' element

array
    '[' ws ']'
    '[' elements ']'

elements
    element
    element ',' elements

element
    ws value ws
```



To format and validate your JSON, just copy + paste it below:

```
{
    "fname": "Rohit",
    "lname": "Katkar"
}
```



APNA COLLEGE

JSON $\longrightarrow$ String

Accessing Data from JSON

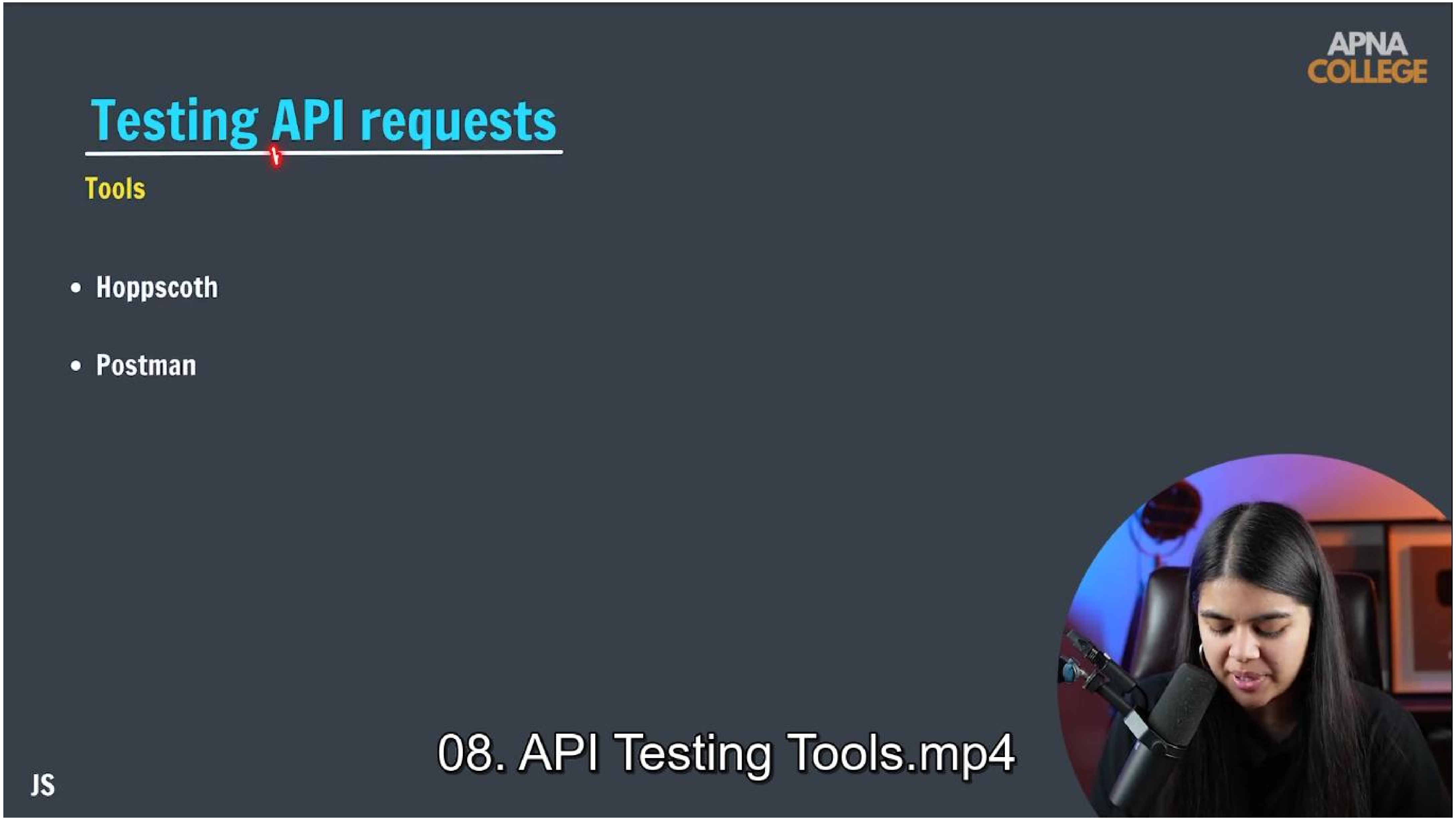
- **JSON.parse(data) Method**

  To parse a string data into a JS object

- **JSON.stringify(json) Method**

  To parse a JS object data into JSON

JS



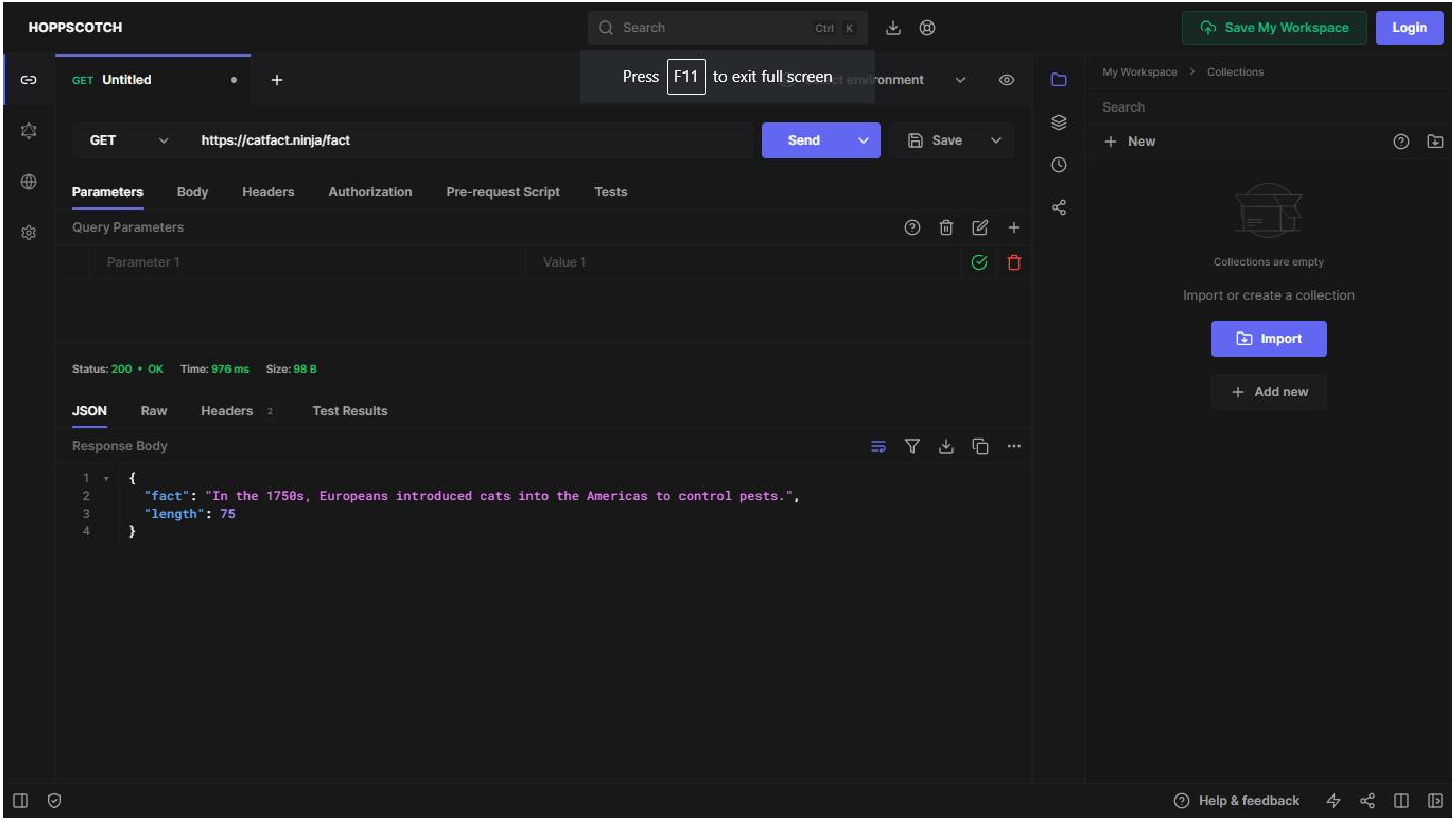
Testing API requests

Tools

- Hoppscoth
- Postman

08. API Testing Tools.mp4

JS



```
{
  "fact": "In the 1750s, Europeans introduced cats into the Americas to control pests.",
  "length": 75
}
```

APNA COLLEGE

Ajax

Asynchronous JavaScript and XML

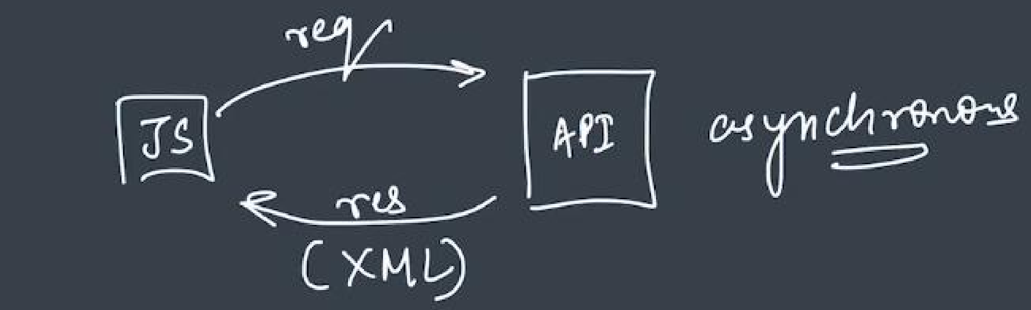
09. What is Ajax_.mp4

JS

APNA COLLEGE

Ajax

Asynchronous JavaScript and XML



JS

Http Verbs

Examples :

- GET
- POST
- DELETE

JS

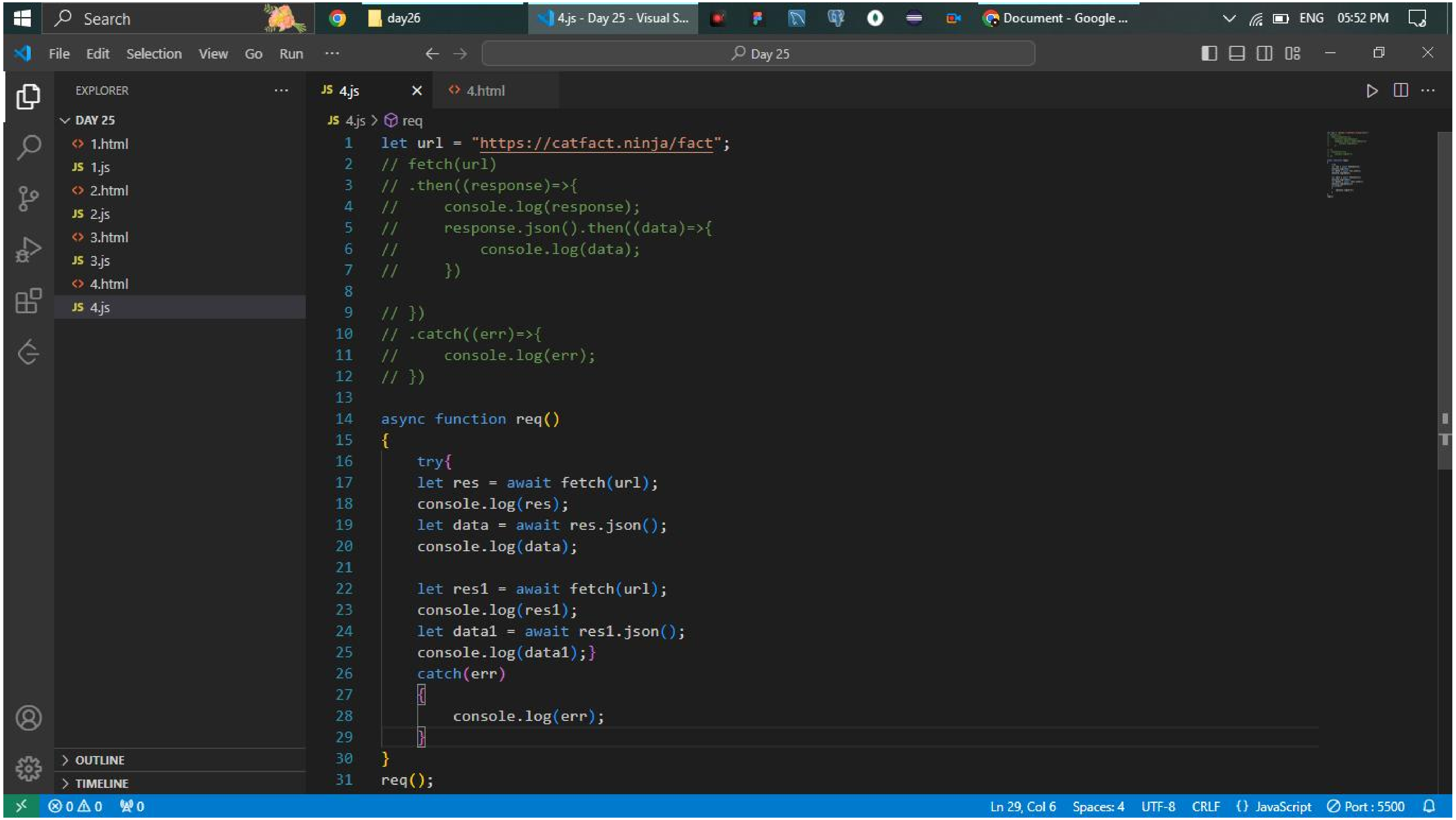
APNA COLLEGE

Status Codes

Examples :

- 200 - OK
- 404 - Not Found
- 400 - Bad Request
- 500 - Internal Server Error

JS



```javascript
let url = "https://catfact.ninja/fact";
// fetch(url)
// .then((response)=>{
//     console.log(response);
//     response.json().then((data)=>{
//         console.log(data);
//     })

// })
// .catch((err)=>{
//     console.log(err);
// })

async function req()
{
    try{
    let res = await fetch(url);
    console.log(res);
    let data = await res.json();
    console.log(data);

    let res1 = await fetch(url);
    console.log(res1);
    let data1 = await res1.json();
    console.log(data1);}
    catch(err)
    {
        console.log(err);
    }
}
req();
```