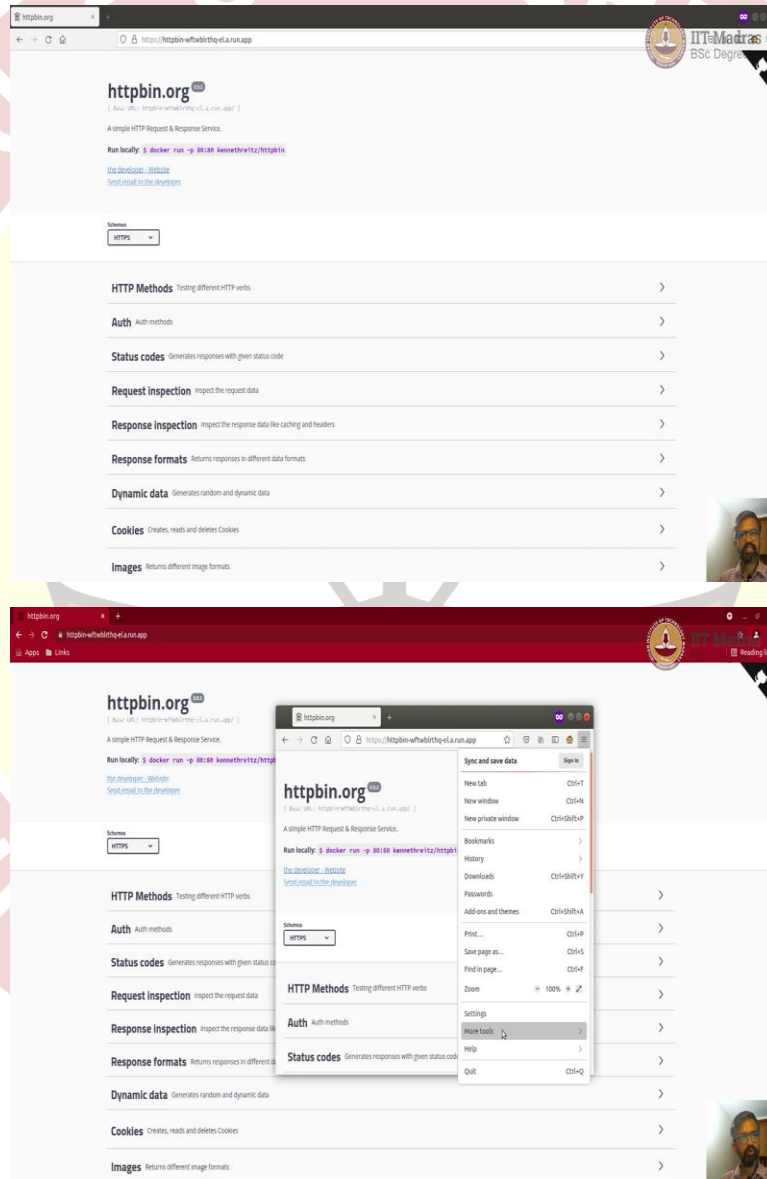
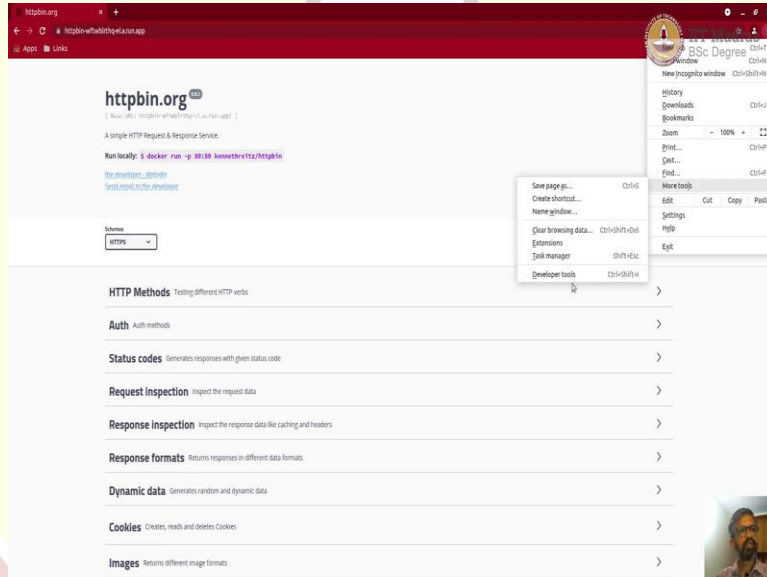
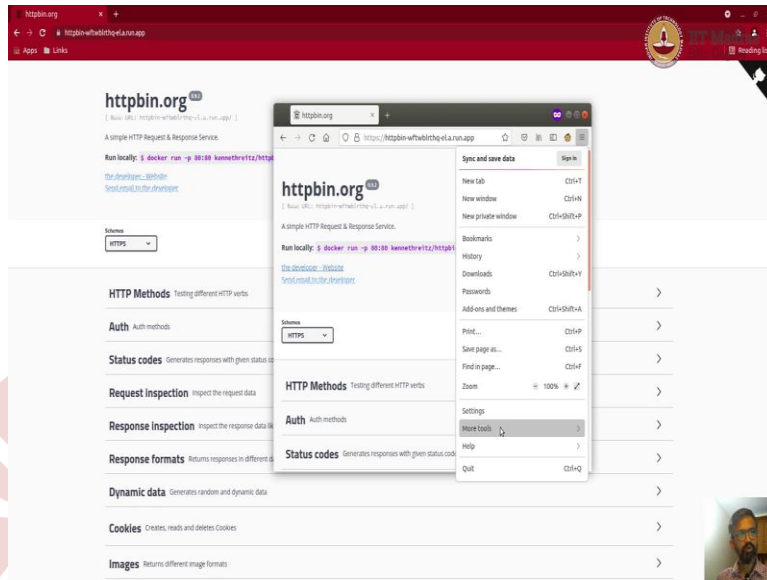


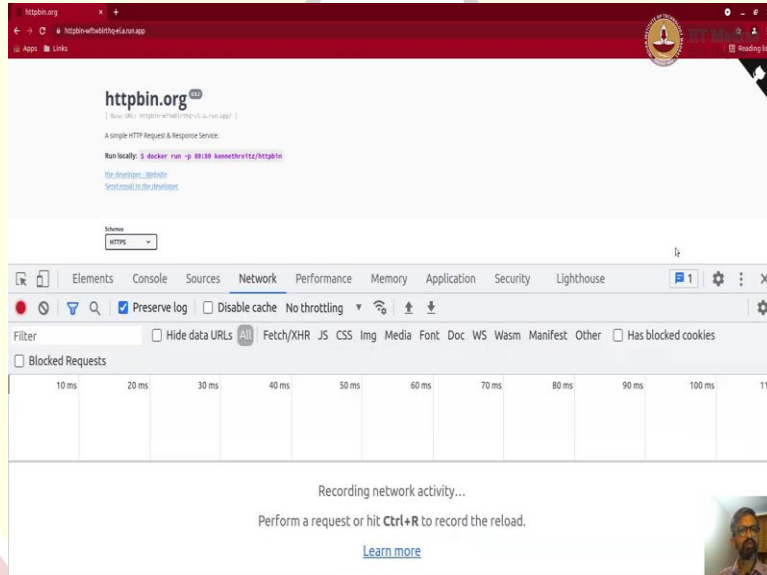
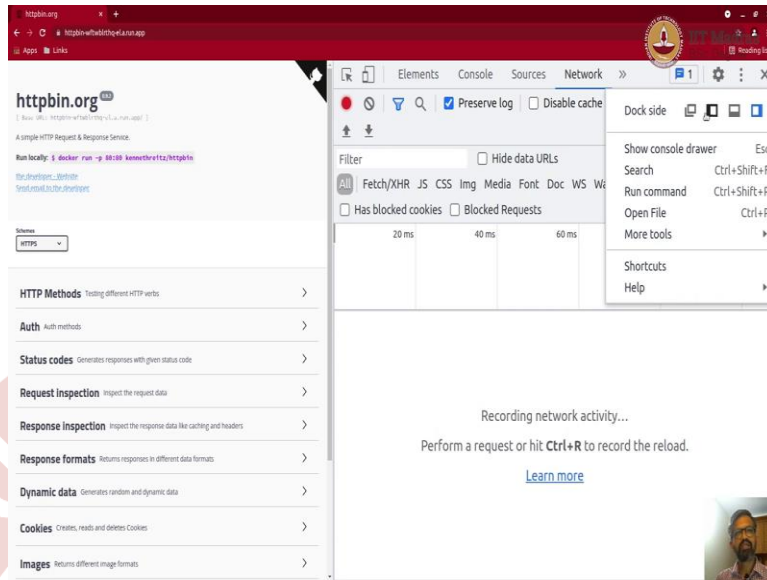
**IIT Madras**  
ONLINE DEGREE

**Modern Application Development - I**  
**Professor Thejesh G N**  
**Software Consultant,**  
**IITM BSc Degree**  
**Indian Institute of Technology, Madras**  
**Basics of Developer tools**

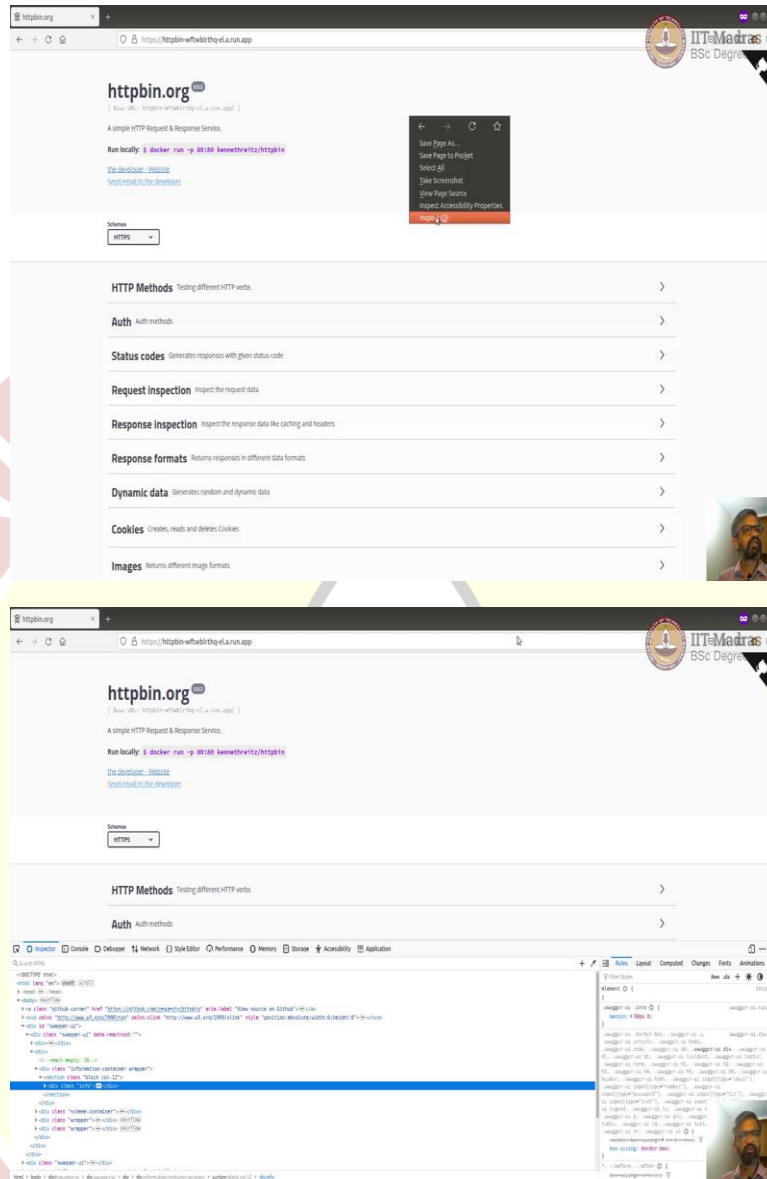
(Refer Slide Time: 00:15)











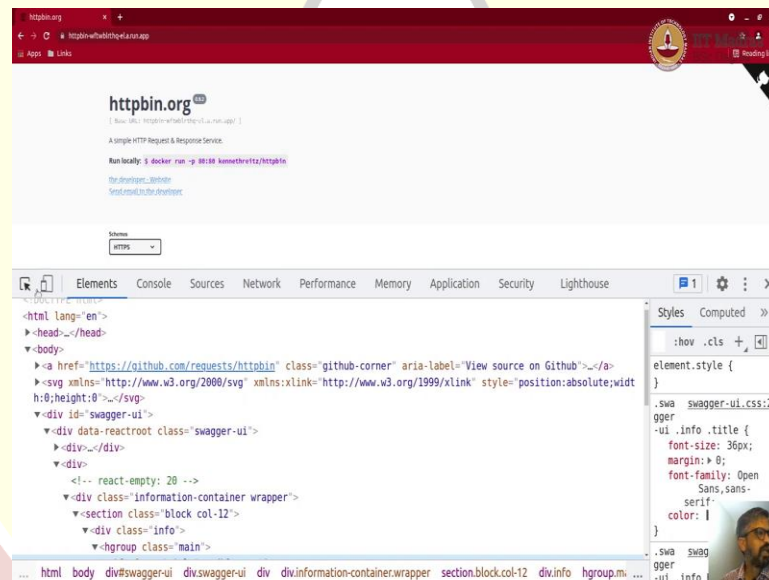
Hi. Welcome to the Modern Application Development screencast. In this short screencast, we will learn how to use browser developer tools or console. What we need are basically a browser, a Firefox or Chrome, both are fine. That is it. So, currently, we are going to use a website called httpbin. It is not, we can use that, but we have our own installation. I will share this URL with you to do our experiments.

How to open a console, that is the first question that one might have. Let us check both in Chrome and Firefox. This is Firefox. You could go to Firefox menu, go to more tools and then you will get many tools here. The first one is web developer tools, click that. And that should open your web developer tools.

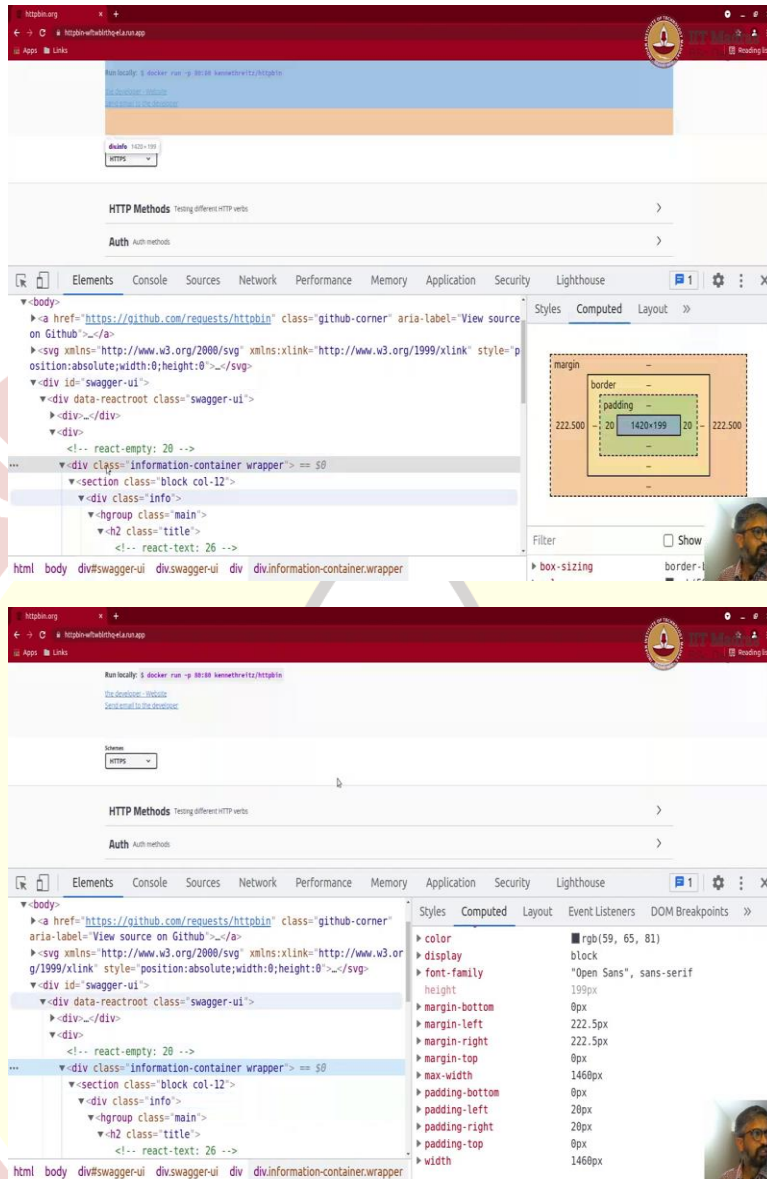
Similarly, in Chrome, you can go to Chrome menu here on the top right, go to more tools, and then say developer tools and that should also open. I have open it on the side here. You can adjust it. You can place it wherever you want by clicking on these three dots and selecting the dock. You can make it come to the bottom and adjust the size. So, this is called developer tools.

The other way to get in here is you can right click on any webpage you want and click on inspect and that should actually open up the developer console. Similarly, in the Firefox too, you can, let me just close this, and right click and click on say inspect and it should open the developer tools or console. Now, I am just going to go through one of the browsers, because they almost look similar. I think if you know the functionalities, you can explore the other one by yourself.

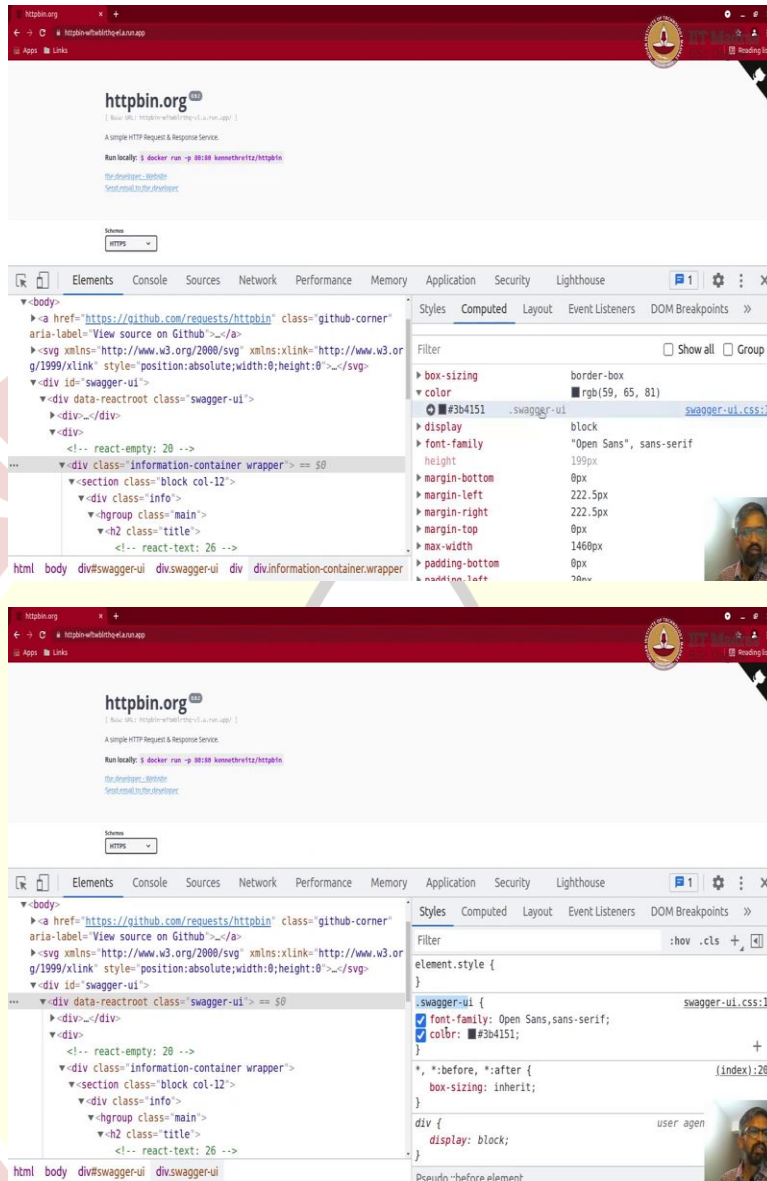
(Refer Slide Time: 02:26)

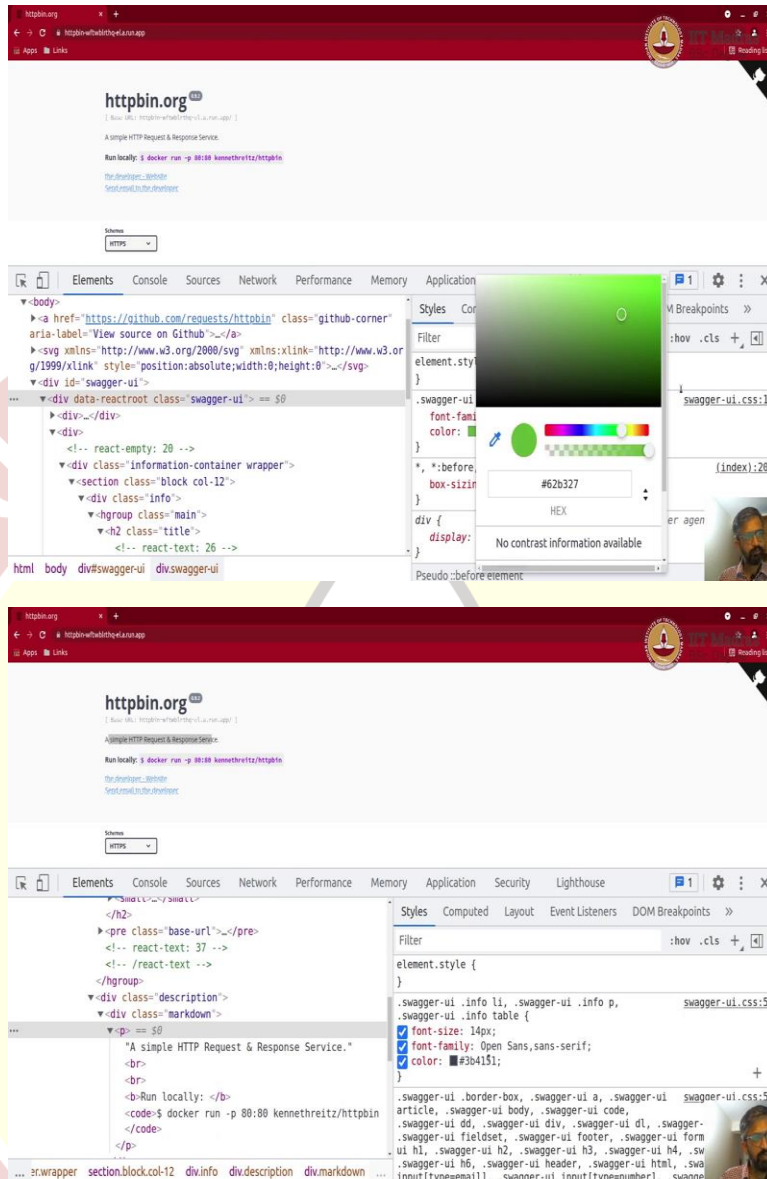




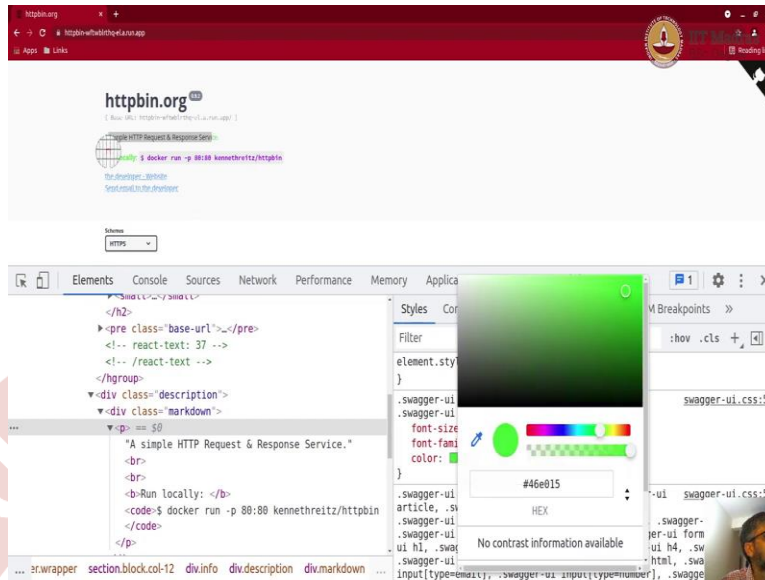








सिद्धिर्भवति कर्मजा



whole style applied is here and which is already applied. So, you can expand them to see how that style is traveled or being applied. Where does it defined and how does it come?

Let us say the color of the font within this whole div is black, this color, almost black or dark grey. If you expand this, say it comes from a based on dot swagger UI which is a class, so you can see that this comes from its parent from this. But the same thing applies because it is a child of this division. So, now, if you click here, this, when you scroll over this, it leads to this arrow. If you click on the arrow, it actually takes to the style. So, you can see that there was a class defined swagger UI, the font families, open sans and sans serif and the color is this.

Now, you can experiment here too. You can switch on and switch off. Or you can change the color too, like, for example, I can change here to a different color to, let us say, green, and it becomes green. And if it is not overridden, then it would become green here. But now currently, it would not get, because it is gotten overridden. Like if I go to here, it is been applied. Let us see. So, I think there is a change. Now, we are at this level, but there is no text at this level.

Let us go to the actual text. There is this. If I right click and say inspect, takes me to that paragraph. That paragraph thing is overwritten. Let us go to computed it of this paragraph. So, it is been overwritten by this color or this style. So, that is where it is a cascading style sheet. So, now, if I click on this, it takes me to a different class, which is, these are the so many styles defined and the same style applies to all of them.

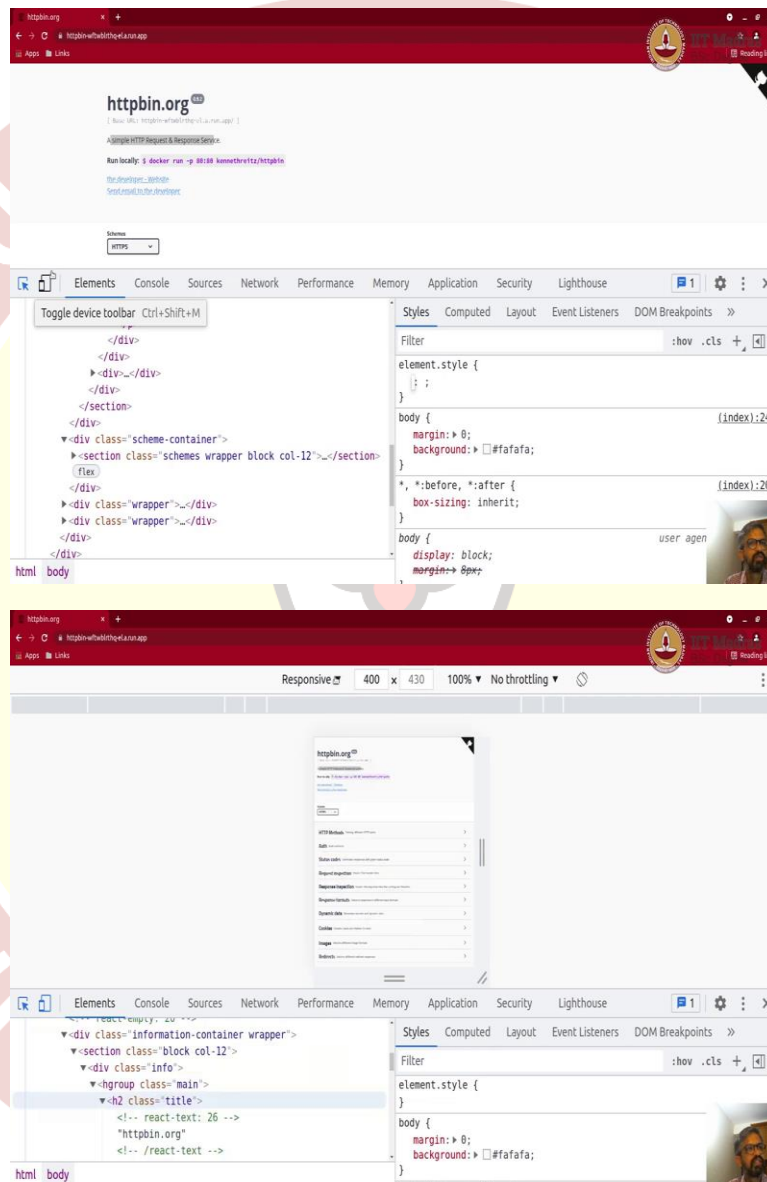
So, swagger-ui .info and li under .info, .swagger-ui, class gain, and .info .p. So, the whole thing is that swagger-ui .info, under .info li under swagger-ui under .info a paragraph. So, that is what is getting applied here, this one.

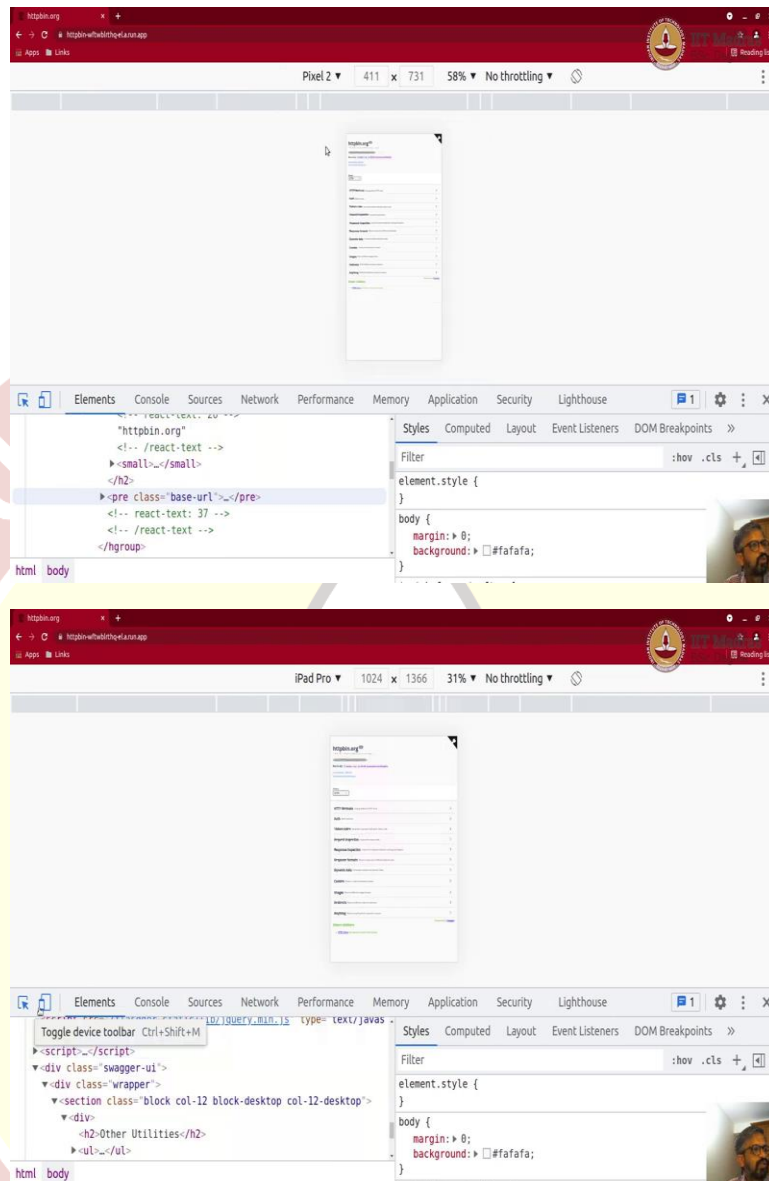
So, now if I change this, let us say, to green, you can see that it is the color has changed here to green. That is the thing. Or if I make it red, it becomes red here. So, this is how we can experiment and try to see whether what style is applying and stuff like that. So, this is both inspect and elements.

You can see the structure of the documents and refer with respect to the browser rendered version here on the top and then see the computed version of the style and go back to what specific style is applied, you can change the style here or you can even apply your own style if

you want to create a new style element. For example, you can click here and create a new style element based on it and then give your own details and experiment like that. So, that is these two tabs.

(Refer Slide Time: 08:39)

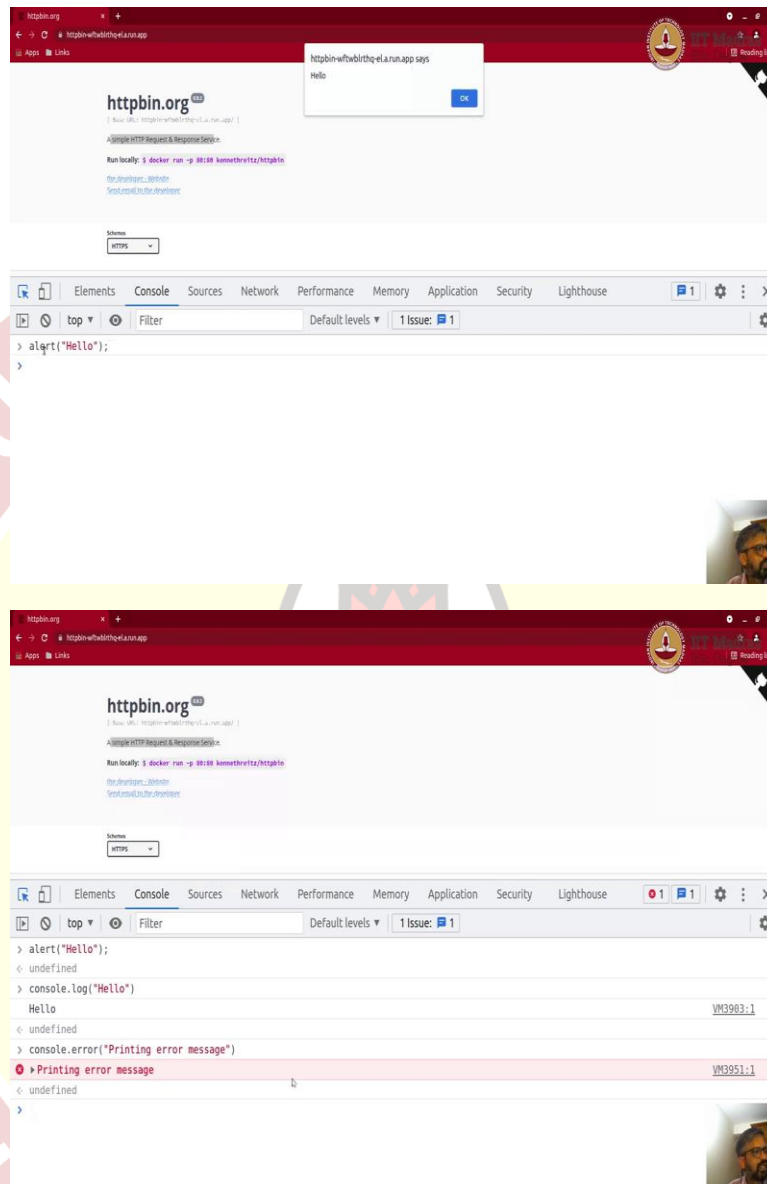




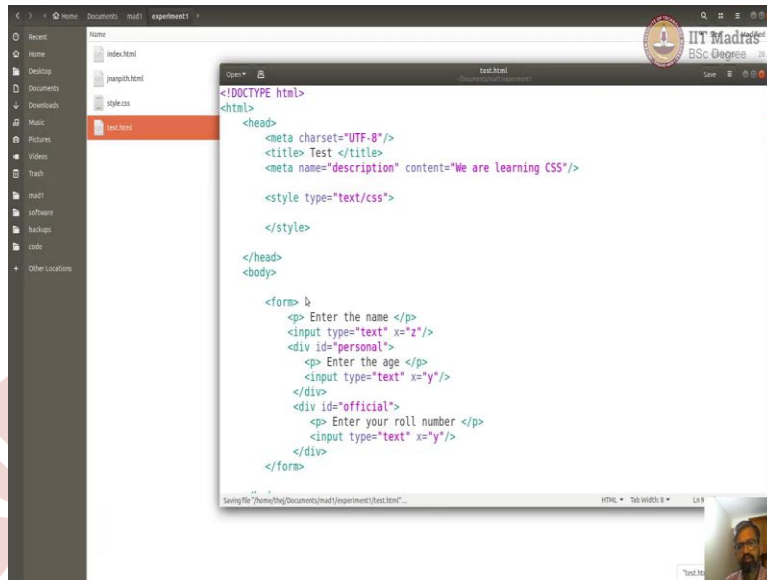
Now, this one here, it shows a toggle device toolbar. So, if you want to see how your web page looks in different types of browsers, or devices, you could use that. Now, this one says it is a responsive screen, but the screen size is this. You can also see how this would look in say in pixel 2, that is how it would look in pixel 2 or let us say iPad Pro, this is how it looks in iPad Pro. So, you can do all that kind of experiment and then see if your page looks nice, does not look nice and stuff like that using this toggle device toolbar.



(Refer Slide Time: 09:21)



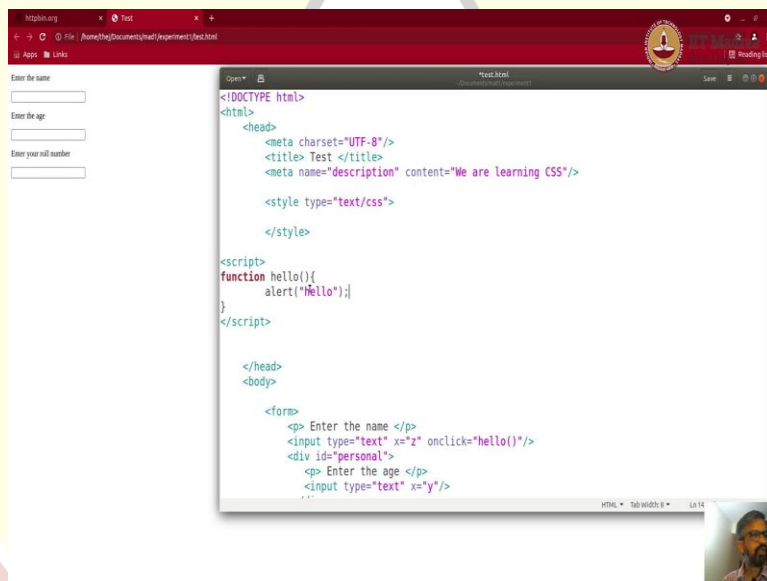




```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title> Test </title>
    <meta name="description" content="We are learning CSS"/>
    <style type="text/css">

    </style>
  </head>
  <body>

    <form>
      <p> Enter the name </p>
      <input type="text" x="z"/>
      <div id="personal">
        <p> Enter the age </p>
        <input type="text" x="y"/>
      </div>
      <div id="official">
        <p> Enter your roll number </p>
        <input type="text" x="y"/>
      </div>
    </form>
  </body>
</html>
```

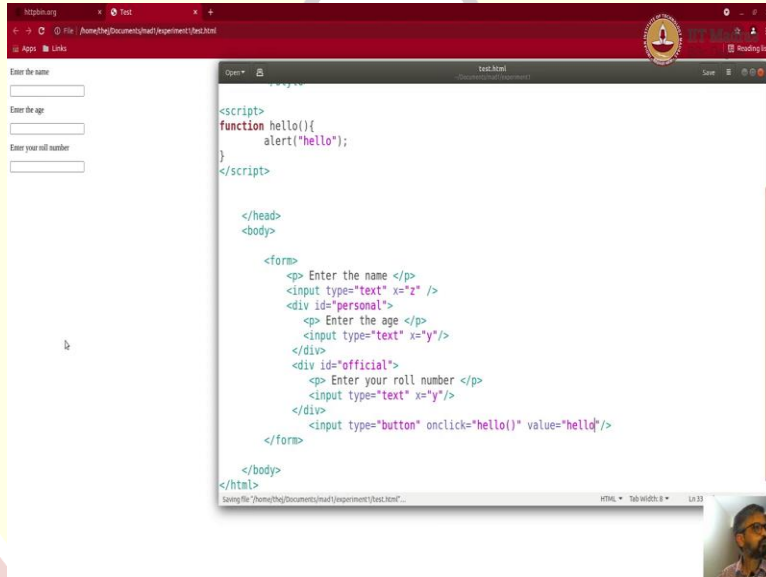
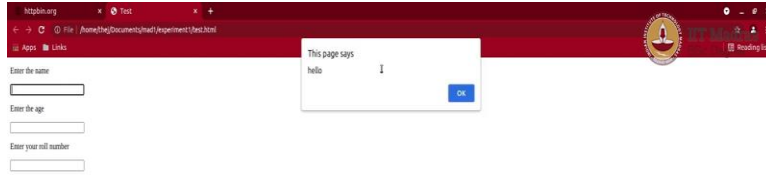


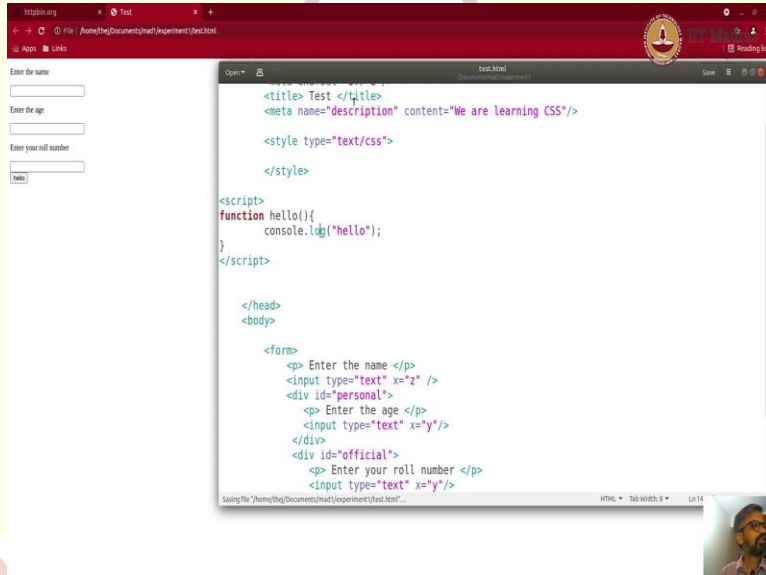
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title> Test </title>
    <meta name="description" content="We are learning CSS"/>
    <style type="text/css">

    </style>
  </head>
  <body>

    <script>
      function hello(){
        alert("Hello");
      }
    </script>

    <form>
      <p> Enter the name </p>
      <input type="text" x="z" onclick="hello()"/>
      <div id="personal">
        <p> Enter the age </p>
        <input type="text" x="y"/>
      </div>
    </form>
  </body>
</html>
```





httpbin.org x Test x +

File /home/thej/Documents/mad1/experiment1/test.html

Apps Links

Enter the name

Enter the age

Enter your roll number

hello

Elements Console Sources Network Performance Memory Application Security Lighthouse

top Filter Default levels No Issues

hello test.html:14

file:///home/thej/Documents/mad1/experiment1/test.html:14

httpbin.org x Test x +

File /home/thej/Documents/mad1/experiment1/test.html

Apps Links

Enter the name

Enter the age

Enter your roll number

hello

Elements Console Sources Network Performance Memory Application Security Lighthouse

Page test.html x

file:///home/thej/Documents/mad1/experiment1/test.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8"/>
5   <title> Test </title>
6   <meta name="description" content="We are learning CSS"/>
7
8   <style type="text/css">
9
10  </style>
11
12  <script>
13    function hello(){
14      console.log("hello");
15    }
16  </script>
```

2 lines, 23 characters selected Coverage: n/a

Watch

Breakpoints No breakpoints

Scope Not paused

Call Stack Not paused

XHR/fetch Breakpoint

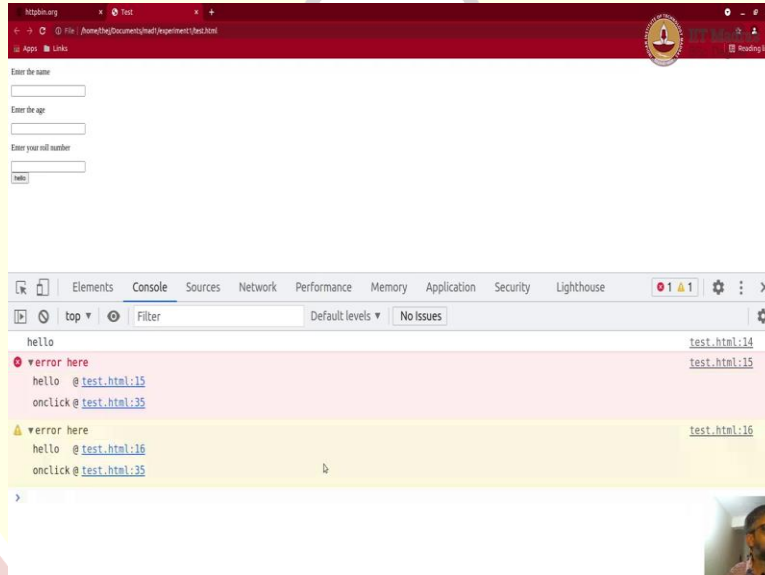
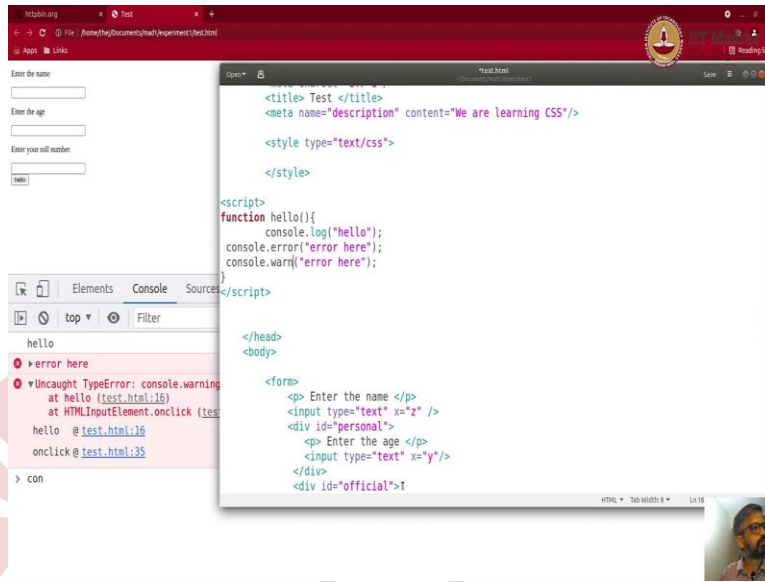
DOM Breakpoints

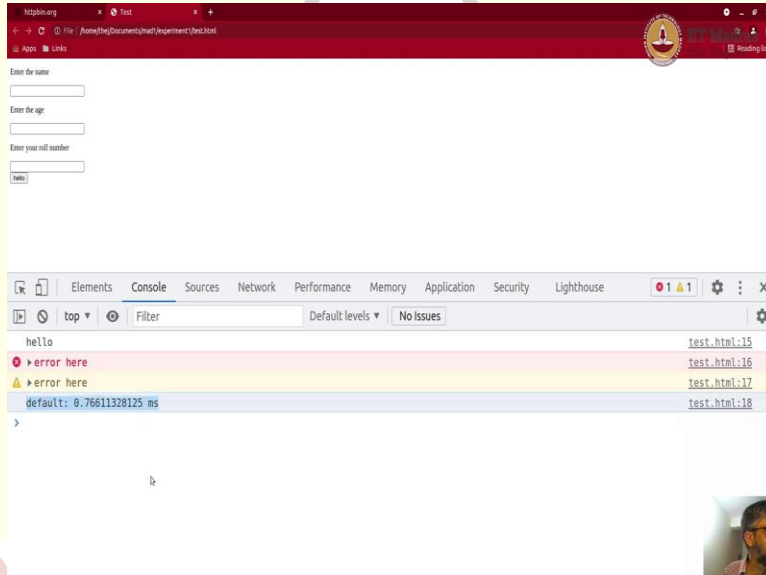
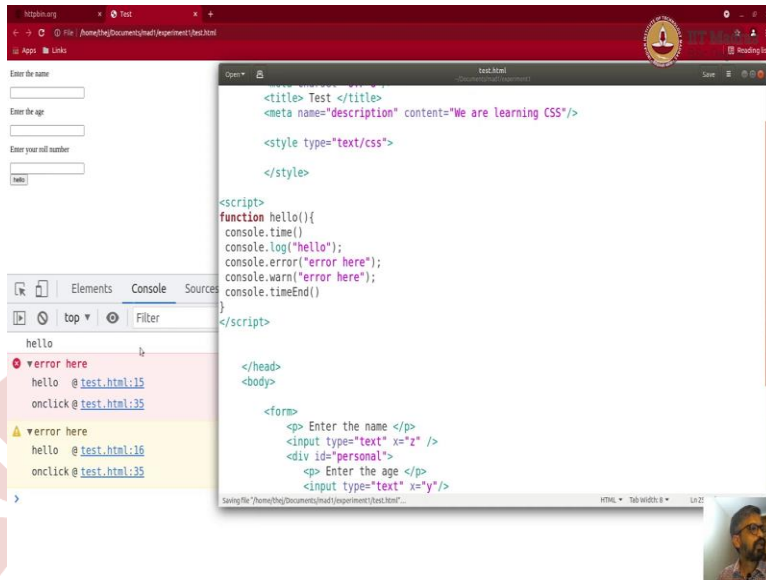
Global Listeners

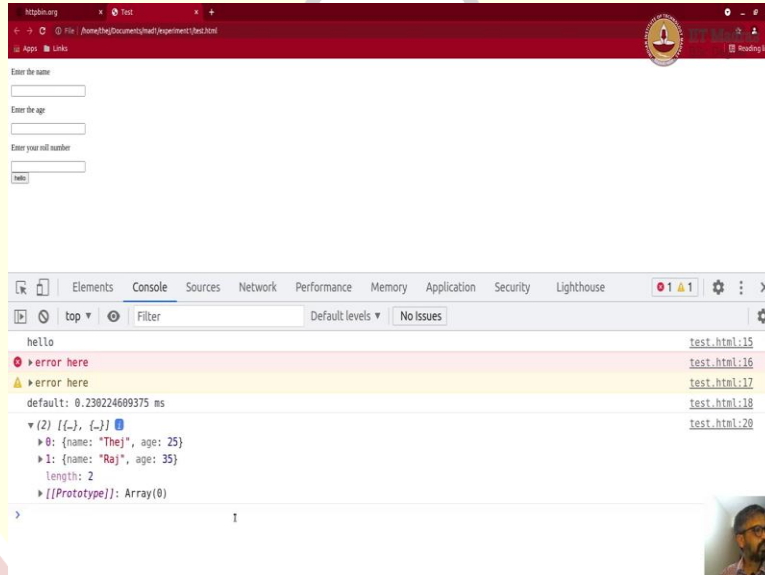
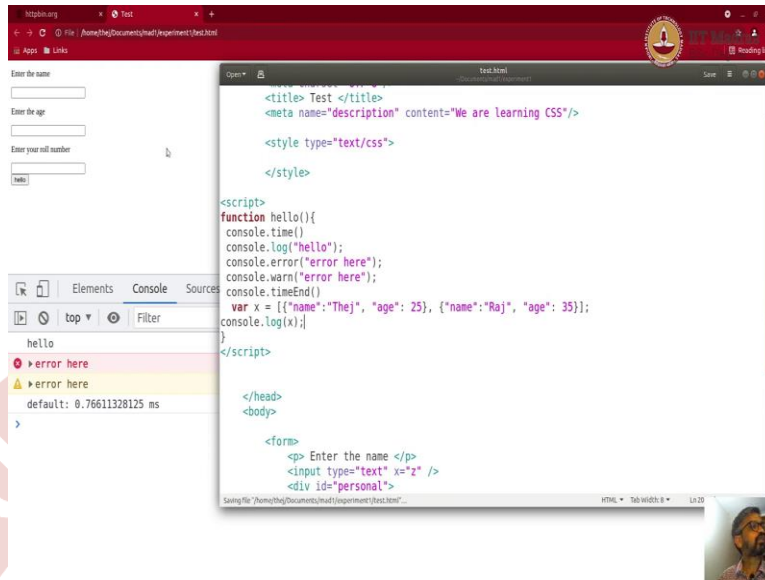
INDIAN INSTITUTE

TECHNOLOGY MADRAS

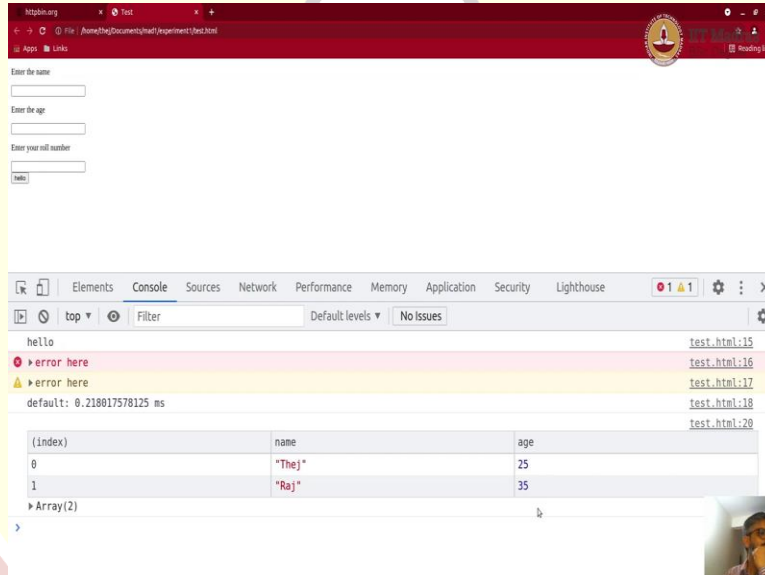
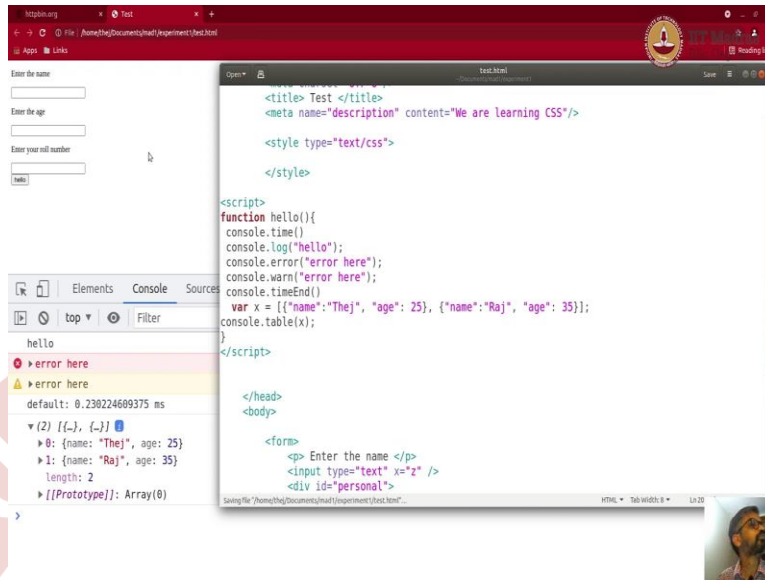
सिद्धिर्भवति कर्मजा

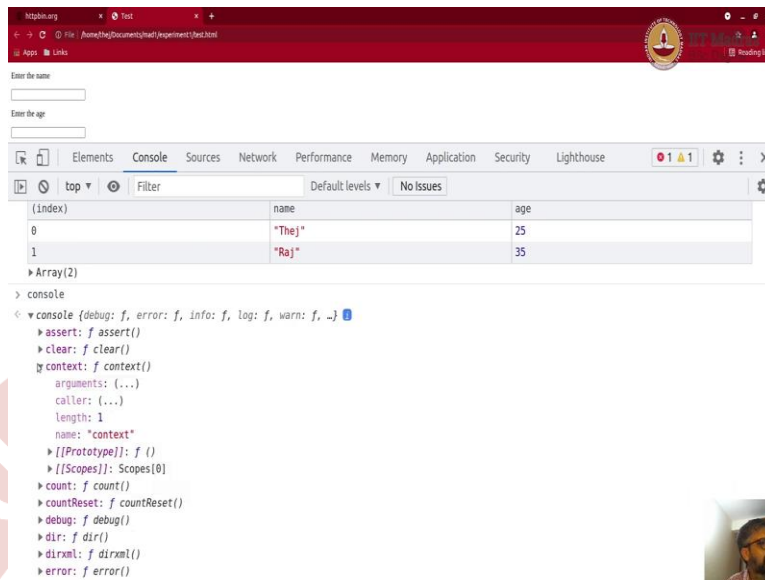












Next is console. Console is like our CLI where if you print something in the program, it prints to the kind of terminal. This is like that. This is like a terminal. You could do something. You can run JavaScript from here, for example, if alert Hello. It actually runs if you press enter and you can see it is running. So, this is like, this is called console, but this is also, you can also run program from here.

So, like many I know one of the best ways to debug is actually print to console. You can do printing your JavaScript variables or attributes or values to console so that you can see them here and it helps to debug. The most often used statement or the log statement is console.log. You can use console.log and you can pass like a message, for example, Hello, it actually prints that. And there are other ones too like console.error to print error, console.warn to print warning, etc.

For example, if I put error, like printing the error message, then it actually prints the error message. Basically, you would not be doing it from here. You will be actually creating a html and printing within that. Let us say, so let us say we have opened an html file which is a simple html file, but we have some JavaScript in it like you are doing some experiments, which runs when the page starts. So, I am just going to add a script tag here and add some JavaScript. It may not, I am not going to do anything significant, but just to show how it works.

Say, pt and let us say you have a function call on click of this textbox, I just want to call hello. So, we will define a function called hello. Now, how do we know this is getting called? We can do an alert and say, hello. So, that is one way of debugging, putting alerts. Let us say if I do this

and if I click on this, it says hello. Let us, instead of putting on a thing, let us put it on a button. It is more intuitive. There is a button. If I click on that it calls hello. But you do not want to do that.

In real world when you are giving and writing an application, you do not want to show like your debugging messages as part, as alert. So, what you could do is console dot log and hello. That would make it, for example, if I open a inspect here, and I will go to console and just click hello, now it does not show the alert, but I still know it went into that part, because I am printing hello. And you can see here on the right side, it shows from where it got printed. If I click on that chose, it got printed from here. So, this will surely help you to debug a lot. This is mostly we will going to be using on and off every time.

Then you can use different kinds of types of logging, sometimes you could, you just want to do console.error. If there is an error and you want to see it in your logs or let us say you want to do warning, you could do warning, change it back to warn. So, if we refresh it, it will go away. I am just going to refresh it again. Click on hello. So, here is the error, here is the warn. Warn actually shows with this yellow and the error shows with red. It also shows actually which you want triggered from where and which line triggered it. So, that is how you debug things.

Now, it can do many other things. It can also say measure the time. If you are doing something like x intensive and you want to measure how much time it took in this JavaScript function to do that, you can start a timer here by saying console.time and you could end the time once you are done with your work, maybe at the end of the function.

Here, now, we are not doing anything significant, but let us assume. We want to know how much time it took to log all of these three statements. I am just going to save. And if I refresh it, and when I run it, you can see that the value or the difference it took, time is 0.76 millisecond from here to here. So, this is very useful if you want to measure the time.

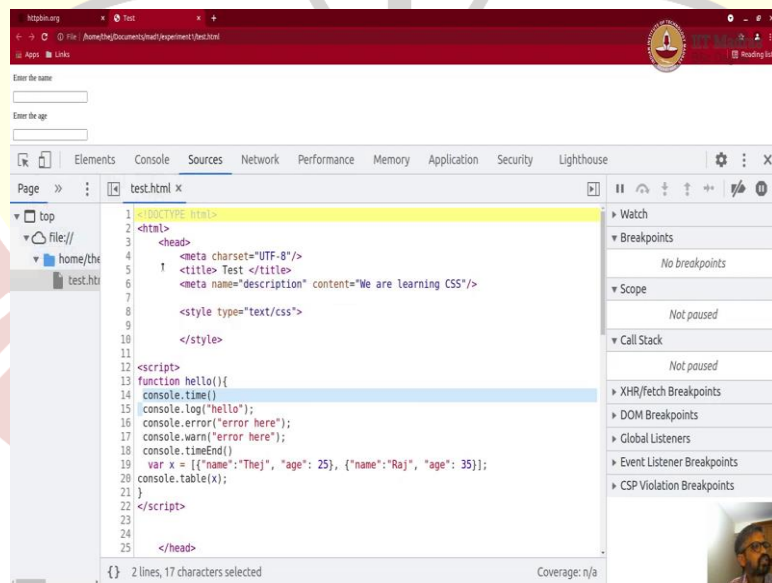
There is another function called console.table, which also helps you print the objects. Let us say you had an object where x is equal to, let us say, we had a name, 25, sorry, and then we had, like, let us say if it was multiples of them. This is, let us say, 35. Let us say his name was, second person's name was Raj.

And one way to see the values of any object is just to log them, and it should be easy x, sorry. So, if I save this, and run this, and click on hello, you can see it is printing. You can explore it, the object and see that each list item or array item, but there is like much more friendlier way to bring this into the console and that is called table.

If you press print table and then let me just refresh it and print hello, you can see the whole object as a table, a whole array as a table. This is much more friendlier and easy to debug than just printing. So, this, you could use it to print any of such content or values. There are many other things like console.trace, console.directory and other things.

You can actually try printing the console and see, like if you type console here and just press enter, it shows all kinds of functions it has. You can see that it has assert, clear, context, count, count reset, debug, directory, each one will tell you details of it, what arguments it takes and you can explore some of them for example. So, you can also do, for example, console.clear. It should clear all the things we have printed until very now. There you go. So that was console and how to use it.

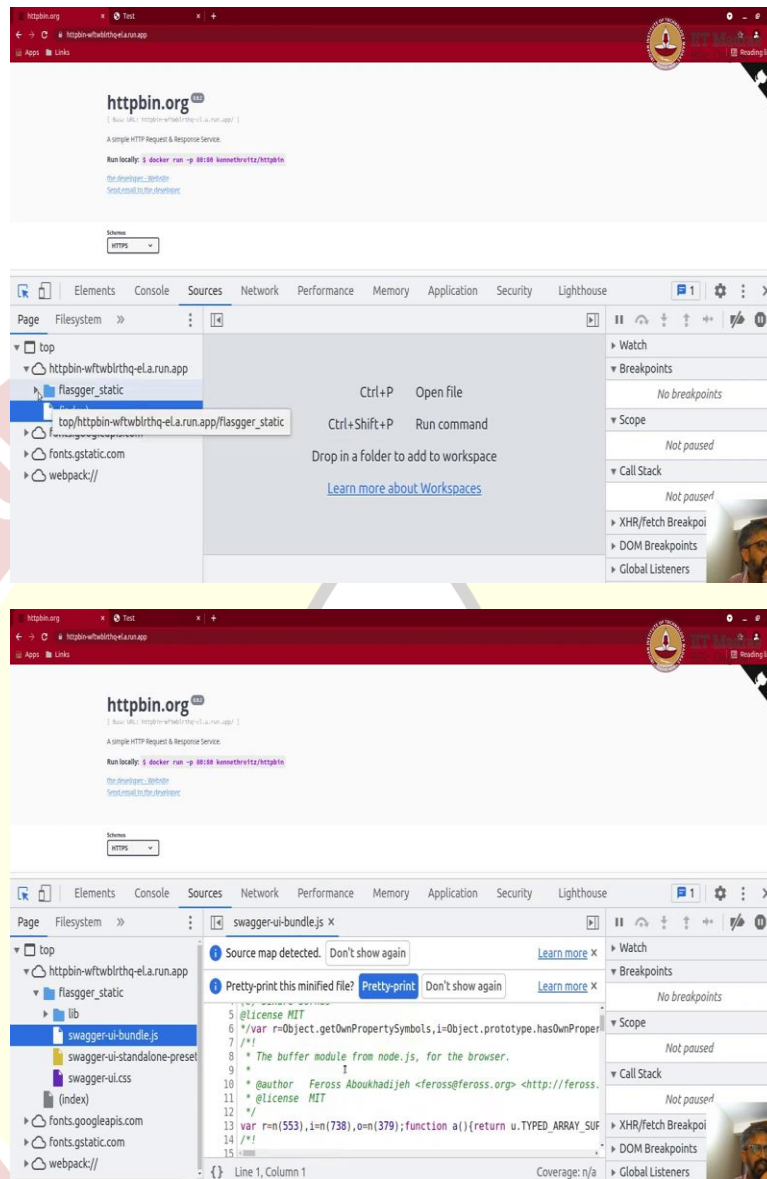
(Refer Slide Time: 18:46)



The screenshot shows a web browser window with a red header bar. Below the header, there are two input fields labeled "Enter the name" and "Enter the age". The browser's address bar shows the file path "file:///home/th.../test.html". The Chrome DevTools interface is open, with the "Sources" tab selected. The "test.html" file is open in the editor, showing the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title> Test </title>
6     <meta name="description" content="We are learning CSS">
7   </head>
8   <style type="text/css">
9   </style>
10
11
12 <script>
13 function hello(){
14   console.time()
15   console.log("hello");
16   console.error("error here");
17   console.warn("error here");
18   console.timeEnd()
19   var x = [{"name":"Thej", "age": 25}, {"name":"Raj", "age": 35}];
20   console.table(x);
21 }
22 </script>
23
24
25 </html>
```

The "Console" tab is also open, showing the output of the code. It displays the time taken for the "hello" function to execute, followed by the log, error, and warn messages, and finally the table of the array x. The table has two columns: "name" and "age". The first row is {"name": "Thej", "age": 25} and the second row is {"name": "Raj", "age": 35}. The "Elements" tab is also open, showing the DOM tree. The "Watch" tab is open, showing the values of the variables in the scope. The "Call Stack" tab is open, showing the call stack of the "hello" function. The "XHR/fetch Breakpoints", "DOM Breakpoints", "Global Listeners", "Event Listener Breakpoints", and "CSP Violation Breakpoints" tabs are also open.

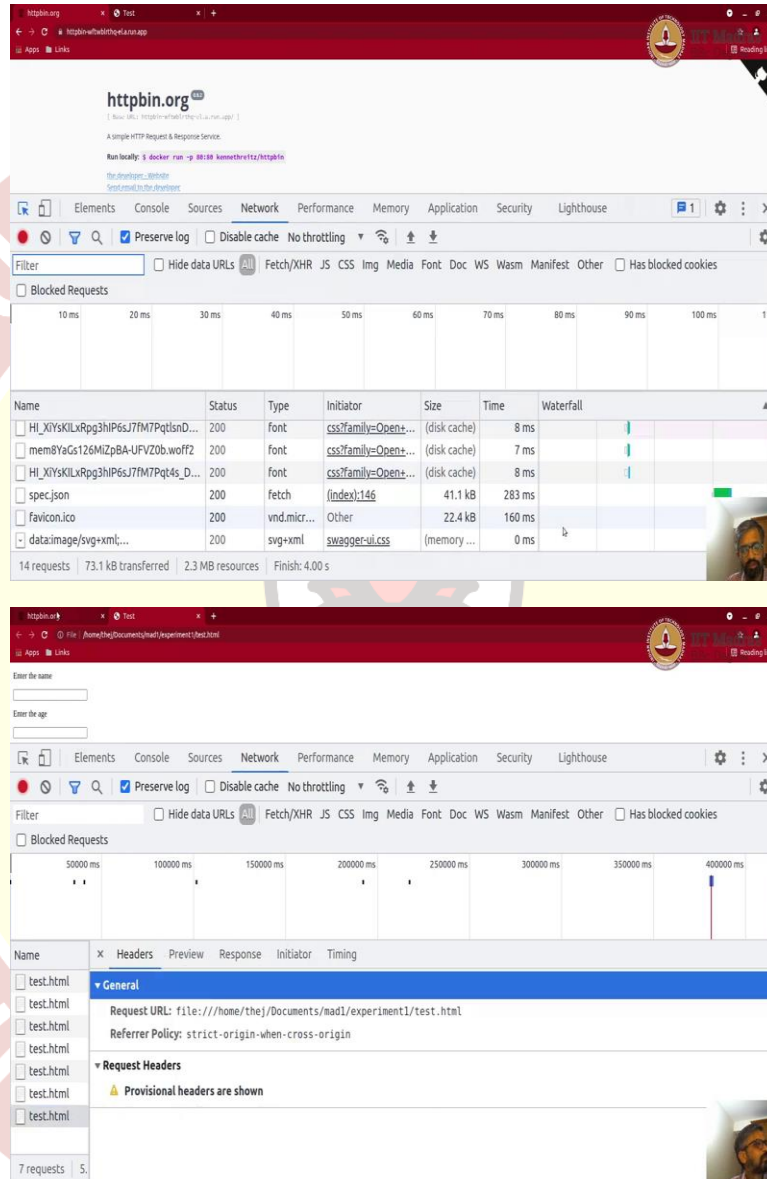


Sources tab is much pretty much simple. It shows content of the html source of it. If there are other contents like JavaScript, CSS, and etc, etc, you could see them too. For example, if I go here, let me just go to the application that we had opened earlier. I am just going to refresh it. Go to source, you can see here different types of sources, like html source, if there are other static images or JavaScript, you can see the source of it. Color formatted with line numbers and stuff like that, you can pretty print as in which makes it more readable and easier to read.

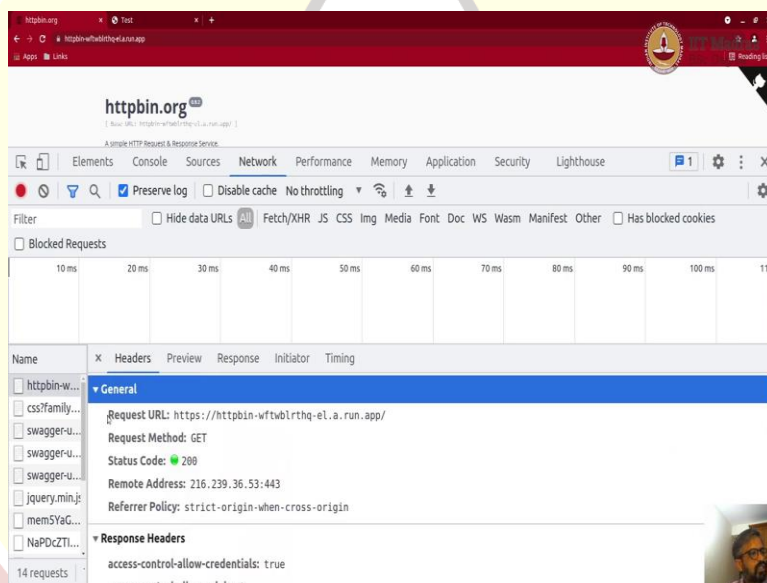
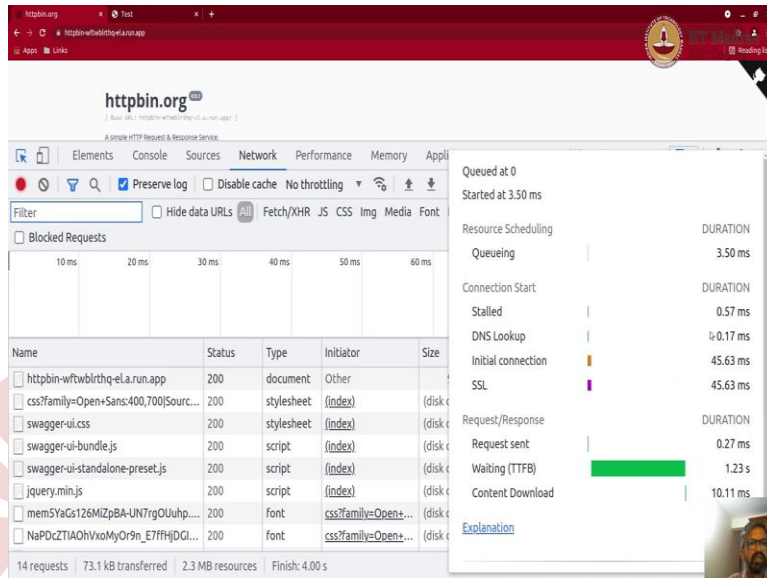
You can also do, I am not going to go through that here, but you can also set up breakpoints. For example, I can set up a breakpoint here and then run through it and pass the execution and all of

that. Not just going to, I think you can keep it as a self learning thing, how to use our debugger inside this sources tab. You can see all the things that has got downloaded.

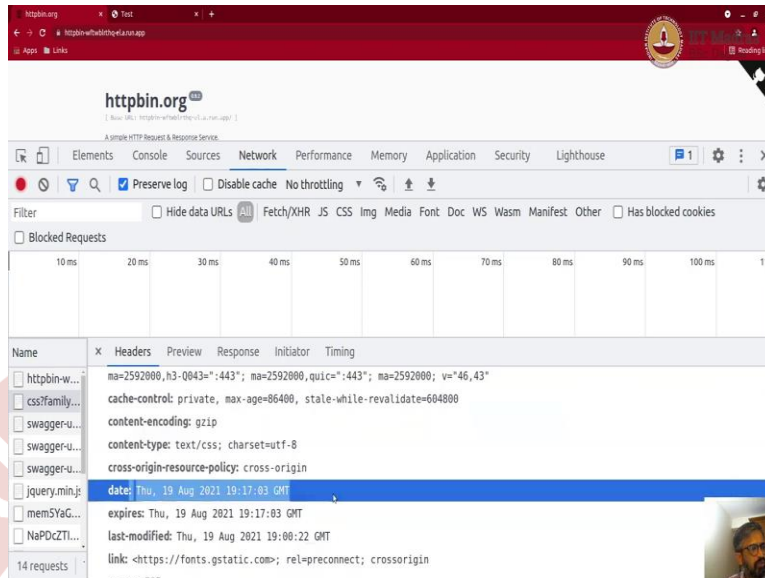
(Refer Slide Time: 19:55)











Next one is a network tab which is much more interesting and very useful. It shows all the network connections that has been made and that has been downloaded and it is getting shown. For example, if I go to this page, there is only one network actually connection, which is loading of html. It shows headers and things like that.

But if it is a complex application, there could be many components. There could be JS, download, CSS, image, other medias, fonts and etc, etc and also Ajax requests. So, you can see here all, to load this page so many network connections has been made and the content has been downloaded only then it got rendered or something.

So, to see each content, how much time it took, you can see in the time column, and the size of the content is size of the downloaded amount of bytes 9.6 kB here. And then you can see the waterfall as well. Waterfall tells you how the whole process went. It got queued first and it got stalled, then did a DNS lookup, got the IP address, and how much time it took, then the initial connection took 45.63 milliseconds, and then SSL, because you need to do key exchange and get the SSL, it took so much, then request was sent, then you waited and then content was downloaded and then it got render.

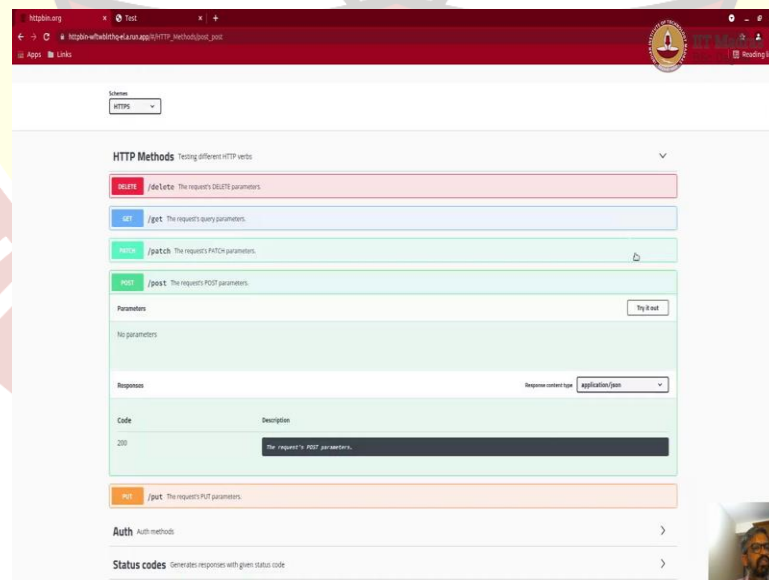
So, this whole thing you can see this 1.29 seconds, how, what all happened that is what you can see in the waterfall. It is the same thing for any kind of content, for example, this took 13 millisecond and we can see what all happened. Similarly, this, in this, you can see it is 1.29

seconds and waiting it was 1.23, stuff like that. You can see the same thing for all kinds of requests.

That is other interesting things you know that when the request is sent there are certain headers are sent and then the response, there are headers and stuff of that. So, any request sent here, you can click on it, it will open a tab next to it. I am just going to close the lower section. So, it opens up a small section here next to it, which shows, the first one is the header. You can see here it shows request is sent to this URL, request type is get, status code is 200 means all okay, good, IP address of the remote connection is this and which port 443 which is https, and any other policy, reference policy. And then this was the request.

And then what are the response headers? So, response headers were set by the server. These were the response headers like interesting things like what is the content size, content type, which it is sending html, if it see the font CSS, then the content type will be CSS, for example, here, text class CSS, and if it is gzip compressed then it will show that and then it will show when it was generated and when it will expire, the caching will expire and many other things. So, this is an interesting things to see. And what all sent and got.

(Refer Slide Time: 23:27)



httpbin.org Test

HTTP Methods Testing different HTTP verbs

**DELETE** /delete: The request's DELETE parameters.

**GET** /get: The request's query parameters.

**PATCH** /patch: The request's PATCH parameters.

**POST** /post: The request's POST parameters.

Parameters: No parameters

Responses: application/json

Code: 200

Response body: {"args": {}, "headers": {"Host": "httpbin.org", "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7; rv:109.0) Gecko/20100101 Firefox/115.0"}, "origin": "192.168.1.1", "url": "https://httpbin.org/post"}}

Elements Console Sources Network

Filter: Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

Has blocked cookies Blocked Requests

200 ms 400 ms 600 ms 800 ms 1000 ms

Name S... T... Initi... S... T... Waterfall

post 2... f... inde... 1... 4... [Green bar]

NaPDcZTl... 2... f... css?... 1... 5... [Green bar]

2 requests 13.5 kB transferred 13.4 kB resources

httpbin.org Test

HTTP Methods Testing different HTTP verbs

**DELETE** /delete: The request's DELETE parameters.

**GET** /get: The request's query parameters.

**PATCH** /patch: The request's PATCH parameters.

**POST** /post: The request's POST parameters.

Parameters: No parameters

Responses: application/json

Code: 200

Response body: {"args": {}, "headers": {"Host": "httpbin.org", "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7; rv:109.0) Gecko/20100101 Firefox/115.0"}, "origin": "192.168.1.1", "url": "https://httpbin.org/post"}}

Elements Console Sources Network

Filter: Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

Has blocked cookies Blocked Requests

200 ms 400 ms 600 ms 800 ms 1000 ms

Name X Headers Preview Response Initiator Timing

post

NaPDcZTl...

**General**

Request URL: https://httpbin-wftwblrthq-el.a.run.app/post

Request Method: POST

Status Code: 200

Remote Address: 216.239.36.53:443

Referrer Policy: strict-origin-when-cross-origin

**Response Headers**

access-control-allow-credentials: true

access-control-allow-origin: https://httpbin-wftwblrthq-el.a.run.app

2 requests 1:

INDIAN INSTITUTE

TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

httpbin.org Test

Apps Links

DELETE The request's DELETE parameters.

GET The request's query parameters.

HEAD The request's HEAD parameters.

PATCH The request's PATCH parameters.

POST The request's POST parameters.

Parameters

No parameters

Execute Clear

Response

Response content type: application/json

Curl

```
curl -X POST https://httpbin.org/post -H 'Content-Type: application/json'
```

Request URL

https://httpbin.org/post

Server response

Code Details

200

Response body

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {}
}
```

Elements Console Sources Network

Filter

Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

Has blocked cookies Blocked Requests

200 ms 400 ms 600 ms 800 ms 1000 ms

Name X Headers Preview Response Initiator Timing

post

NaPdCZTL...

args: {}, data: "", files: {}, form: {}...

headers: {Accept: "application/json", Accept-Encoding: "gzip, de...

origin: "49.287.222.180"

url: "https://httpbin-wftwblrtq-el.a.run.app/post"

2 requests 1

httpbin.org Test

Apps Links

POST The request's POST parameters.

Parameters

No parameters

Execute Clear

Response

Response content type: application/json

Curl

```
curl -X POST https://httpbin.org/post -H 'Content-Type: application/json'
```

Request URL

https://httpbin.org/post

Server response

Code Details

200

Response body

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {}
}
```

Elements Console Sources Network

Filter

Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

Has blocked cookies Blocked Requests

200 ms 400 ms 600 ms 800 ms 1000 ms

Name X Headers Preview Response Initiator Timing

post

NaPdCZTL...

Request initiator call stack

1 {

2 "args": {},

3 "data": "",

4 "files": {},

5 "form": {},

6 "headers": {

7 "Accept": "application/json",

8 "Accept-Encoding": "gzip, deflate, br",

9 "Accept-Language": "en-GB,en-US;q=0.9,en;q=0.8,hi;q=0.7"

10 "Content-Length": "0",

11 "Forwarded": "for=49.287.222.180;proto=https",

12 "Host": "httpbin-wftwblrtq-el.a.run.app",

13 "Origin": "https://httpbin-wftwblrtq-el.a.r

14 "Referer": "https://httpbin-wftwblrtq-el.a.r

15 "Sec-CH-UA": "Chromium";v=103.0"

2 requests 1: Line 1, Column 1

सिद्धिर्भवति कर्मजा

Network tab: POST /post. The request's POST parameters.

Parameters: No parameters.

Response: application/json

Request URL: https://localhost:3000/post

Server response: 200

Response body: [{"id": 1, "name": "John", "age": 30, "email": "john@example.com"}, {"id": 2, "name": "Jane", "age": 25, "email": "jane@example.com"}]

Request call stack:

- (anonymous) @ index.js:1
- X @ runtime.js:62
- (anonymous) @ runtime.js:296
- e.<computed> @ runtime.js:114
- r @ asyncToGenerator.js:17
- (anonymous) @ asyncToGenerator.js:28
- Promise.then (async) @ asyncToGenerator.js:27
- r @ asyncToGenerator.js:35
- (anonymous) @ export.js:36

Network tab: POST /post. The request's POST parameters.

Parameters: No parameters.

Response: application/json

Request URL: https://localhost:3000/post

Server response: 200

Response body: [{"id": 1, "name": "John", "age": 30, "email": "john@example.com"}, {"id": 2, "name": "Jane", "age": 25, "email": "jane@example.com"}]

Timing:

Event	Duration
Resource Scheduling	
Queueing	1.21 ms
Connection Start	
Stalled	0.33 ms
Request/Response	
Request sent	
Waiting (TTFB)	
Content Download	

सिद्धिर्भवति कर्मजा

[illegible]



https://httpbin.org/post

```
{
  "args": {},
  "headers": {
    "Host": "httpbin.org",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36",
    "Accept": "application/json",
    "Content-Type": "application/json",
    "Content-Length": "124",
    "Accept-Language": "en-US,en;q=0.9",
    "Cache-Control": "no-cache",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36",
    "X-Cloud-Trace-Context": "6a2b6748b6b6c34302056cece4d91037f913227f80271c42"
  },
  "origin": "192.168.1.1",
  "url": "https://httpbin.org/post"
}
```

Elements Console Sources Network Performance

Filter: ☐ Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

☐ Has blocked cookies ☐ Blocked Requests

50 ms 100 ms 150 ms 200 ms

Name	X	Headers	Preview	Response	Initiator	Timing
post						

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36

Form Data view source view URL-encoded

custname: Thejesh

custtel: 9999900000

custemail: i@thejeshn.com

size: large

topping: cheese

delivery: 11:15

comments: Nothing

Customer name: Thejesh

Customer email: i@thejeshn.com

Customer address: i@thejeshn.com

Pizza Size: ☐ Small ☒ Medium ☐ Large

Pizza Toppings: ☐ Bacon ☒ Extra Cheese ☐ Onion ☐ Mushroom

Preferred delivery time: 11:15

Delivery instructions: Nothing

Submit order

Elements Console Sources Network Performance

Filter: ☐ Hide data URLs

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

☐ Has blocked cookies ☐ Blocked Requests

50 ms 100 ms 150 ms 200 ms

Name	X	Headers	Preview	Response	Initiator	Timing
post						

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36

Form Data view source view URL-encoded

custname: Thejesh

custtel: 9999900000

custemail: i@thejeshn.com

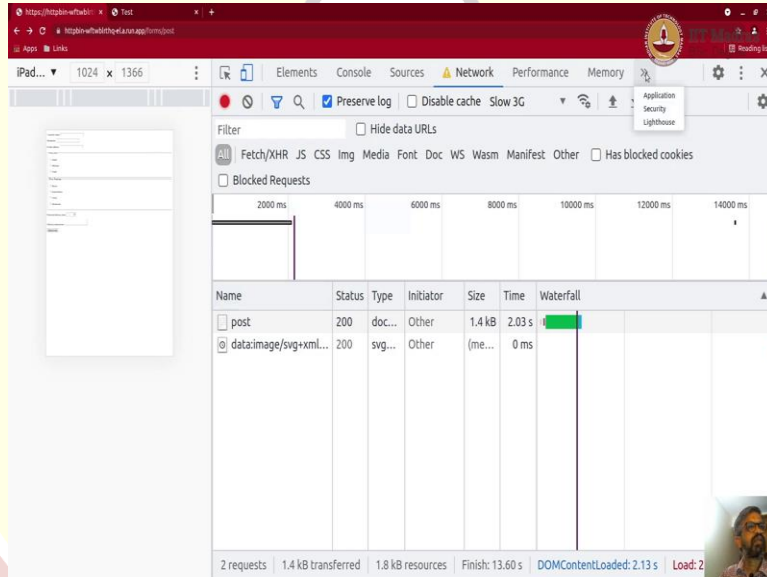
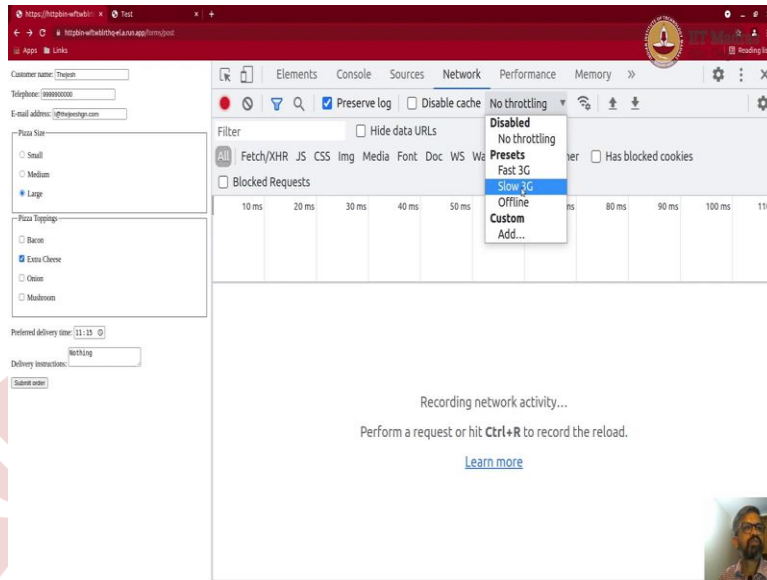
size: large

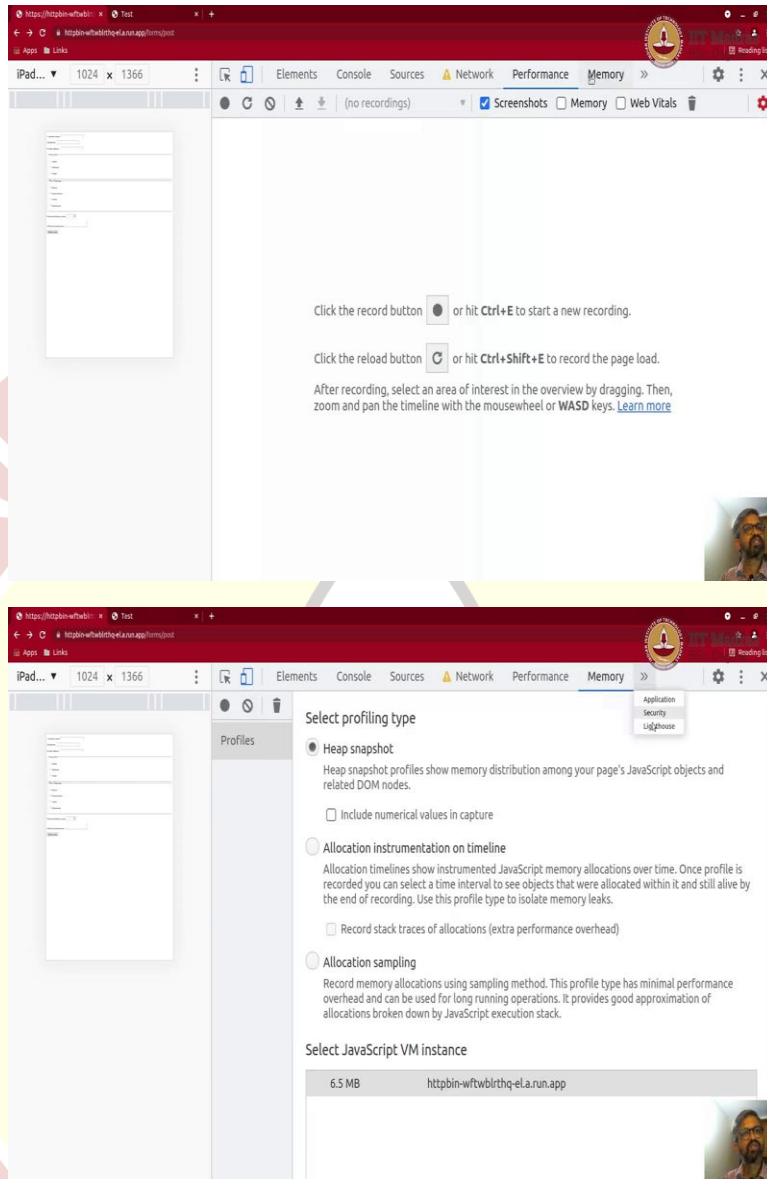
topping: cheese

delivery: 11:15

comments: Nothing







Now, let us do like one of the post or get request, let us do one post request and see how it gets sent. I am going to open inspect. I am going to open it on the right side to make it easy to view like that. I am going to go to network. I am going to be in this section when I make a post call. When I click on try it then click on execute which will send a post request to this server. Click on request.

So, you can see that it sent a post request which is equivalent of this kind request, let us say, it is, here you can say it is post request and request details are here post 200 and stuff, response header, request headers and then you can see preview the response. This is the data it got sent. You can see that response has all of this. It is the same thing here.

If you want to see non-previewed version, the raw content you can see here. Who initiated this call, it is this page and the timing part of it like we saw before. Now, one more thing is here, we are saying headers. We are not actually sending any data. As you can see, we are not sending any data. We could actually send some data and see them here too.

Let us try like another experiment where we are going to send some data. Let us say we are doing a form submission. So, here is a form. I am just going to clear this section. Here is a form. Let us fill up some name, Thejesh, telephone is 9999900000, my email is i@thejesh.com. I am just going to ignore and put some random values for the rest.

Now, when I submit it does a post request here. So, let us do, what did it say? It should be post current time. So, it made a post request. You can see here. So, you can see the request type is post, status is 200. The most interesting part is if you go down, you can see the form data that has been sent to the server.

This is very useful when you are trying to debug what has been submitted, what has not been submitted. You can see that it has sent custom name is Thejesh, custom telephone is 999 and all of them. You can also see the very raw version of it like this or a past version. So, you can see custom name.

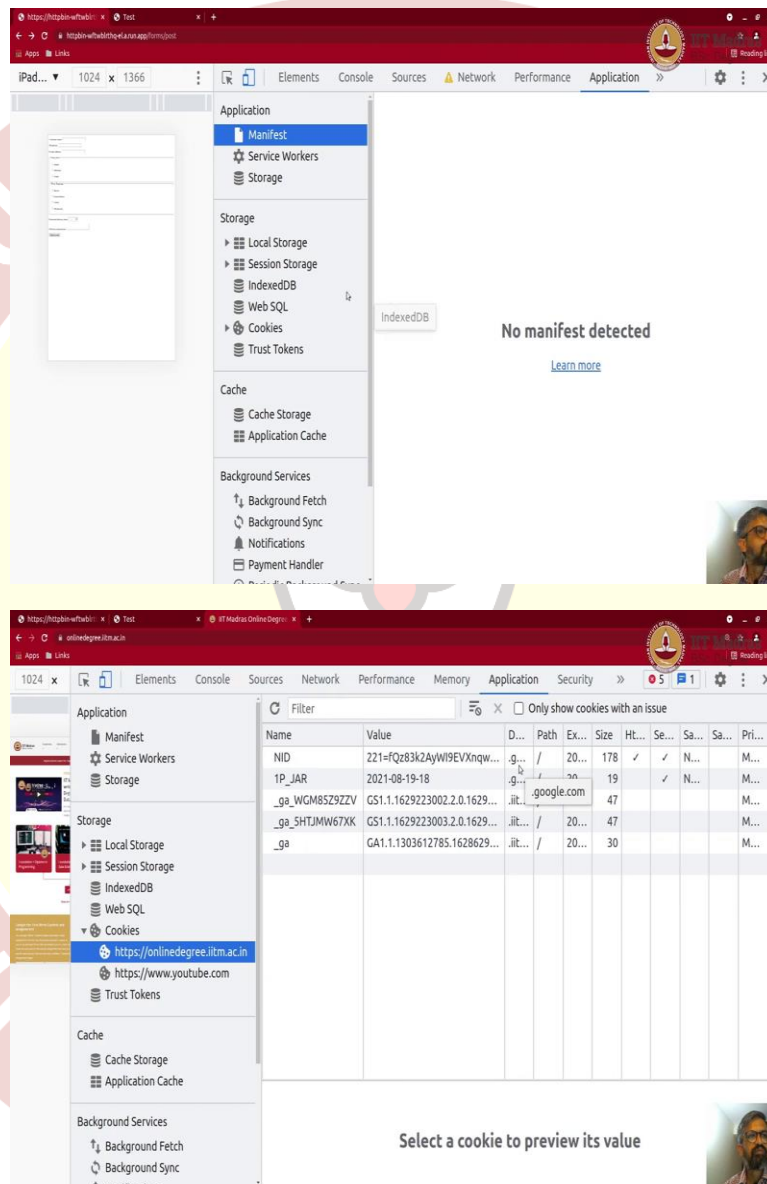
So, this is name of the form element and this is the value, like for example, custom email. If I go back here and if I view source, inspect, see here, it is the custom email this name. That is the name of the form element that is been sent. It would not send the type it sends with the name. So, that is the value that is been sent. And that is how we use network tab to debug the things. There are many other things that you can do. You can preserve the log across the request. If you do not click on this, if you make the next request, the current request will vanish.

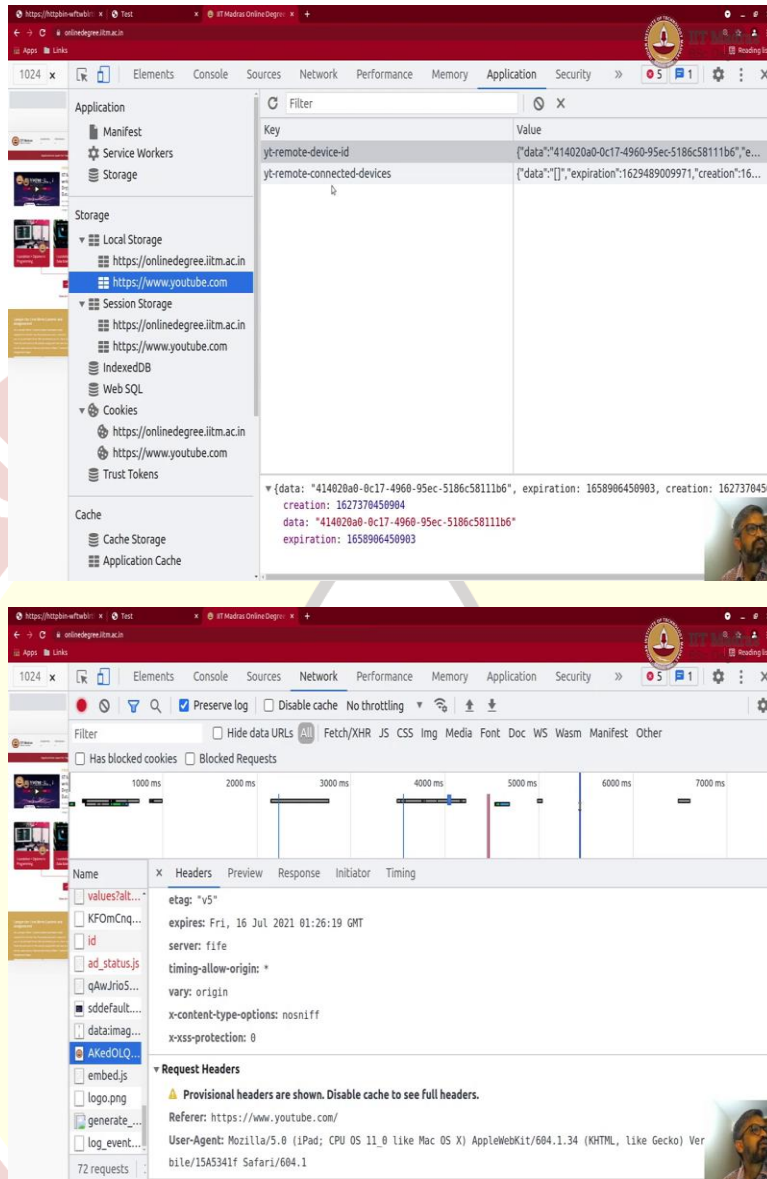
This is how usually I enable preserve the log, and whenever I want to clear, I will clear. There is other things. You can also simulate, like request based on different mediums of transport. For example, if you are accessing a website on mobile, how fast will it be? You can throttle it to that level.

For example, I can say just slow 3G and if I do refresh slow 3G, it will take some time, it would not be that fast. Things like that you can simulate and see how it works. So, there are two

simulations. One is the network bandwidth simulation and then there is how the page looks on the mobile simulation. So, those are two important things. In the other tabs, you can explore yourselves the performance tab or the memory tab.

(Refer Slide Time: 28:41)





I just go to application tab because this is important, especially for the content that you store locally in the local session or the local storage. You might want to clear them sometimes to just debug the things. So, you can access local storage and any values that it has. Usually, it will be key value pair.

Similarly, you can access session storage. It will be here if you want access. Similarly, indexedDB, it is like a local DB. You can see access the DB here in this section and clear them. Web SQL if it is enabled. Not all browsers actually support web SQL. And then there is cookies. If there are any cookies set up you can see them here.

Let us take some other site. If I go to this site, right click inspect and if I go to application and see you can see that there are two cookies set one by online degree there are some cookies set and there are some cookies set by YouTube. So, you can see the values of it and where it is come from, source of it and all of it.

So, you can see the source of it is iitm.ac.in, domain of it, this has come from google.com. And this is the key and this is the value and the size of the cookie, many other attributes. If you want to clear them, you can actually clear them too. You can just delete cookies.

Sometimes when you want to clear the whole session, you could do that. Similarly, you can check session storage if you kept any values. There is no values here, similarly, here. So, YouTube has stored some session values or in the local storage. If you want to clear it out, you can delete that thing.

Or you can even a like if you are trying to do some experimentation and want to edit, you can edit here. You can keep many into JavaScript objects in this. It is quite interesting. I think you should explore the local storage and session storage. Actually, you, if the document has cached anything, it will be in the cache storage.

So, if you go to your network, and let us say if I refresh this page, so some of it might show where it is, some of them may not get loaded again, because it would have been already cached. So here it is. It is served from the local disk cache. So, there may not be any network traffic for this because it is actually from the local cache, and we do not need to get it again. So, that also you can see. So, these are the interesting things that you can see using the developer console.

Now, you should understand that web is made for creators. Browser is as much as a tool for developer as much it is for an explorer of the web. If you are a web developer, there is whole set of tools under developer console to use for daily basis and it is a lot and have very less time to cover it. I think you should explore by yourself. Spend couple of hour, I am telling you it is worth your time. Thank you for watching this episode. Bye.