

Introduction to Modern Application Development
Prof. Tanmai Gopal
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - P4
Lecture - 10
Server Side Programming and Introduction to HTML and CSS

Hi all. Welcome to module - P4. In this module, we will be adding a few more pages to our web app and by doing that exercise we will learn little bit about server side programming and getting an introduction to HTML and CSS.

(Refer Slide Time: 00:13)



The slide displays the following content:

Contents

Objective:

1. Understand server-side programming to make different URLs
2. Intro to HTML, CSS
3. Intro to server-side templating

Topics:

1. URL based routing
2. Creating new HTML pages
3. HTML, CSS fundamentals
4. <a> links: The Real Thing That Makes The Internet Work

At the bottom of the slide, there is a footer with the text: "Introduction to Modern Application Development" on the left and "Dr. Gagan Raju (IIT Madras), Tanmai Gopal (IIT Madras)" on the right. A small video inset in the top right corner shows Prof. Tanmai Gopal.

Our objective in this course is to understand how server side programming works and how we make different URLs and handle those different URLs. We will also try to learn the fundamentals of HTML and CSS. And we will be introducing the concept of server side templating which in the future we will lead as onto databases.

(Refer Slide Time: 00:33)



My Programming Task

I want to make a few simple web pages

I want those web pages to be published and available on the internet and accessible at certain URLs

Introduction to Modern Application Development | Dr. Ganesh Rajan (BIT Madras), Tamasz Csepel (Hasura)

This entire module is structured around a very simple task. The task is that we want to make a few simple web pages, and we want to publish those pages on our web app by making it accessible on certain URLs.

(Refer Slide Time: 00:46)



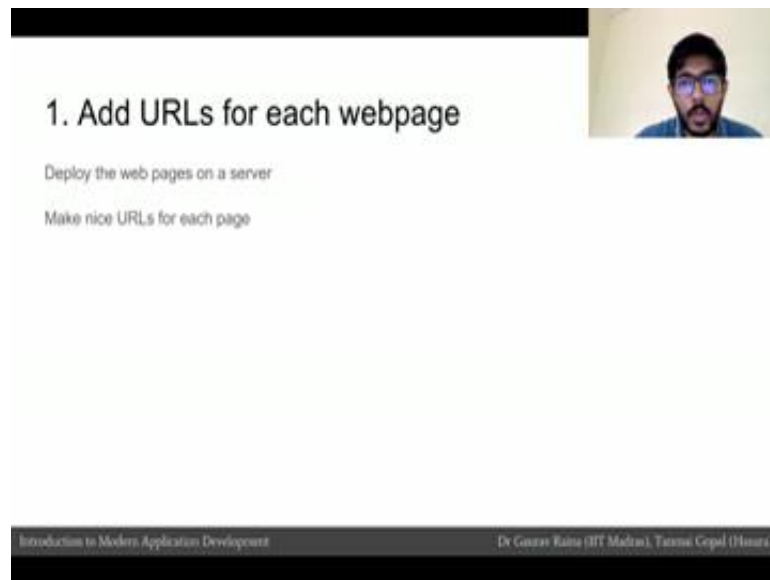
Our final target

URL	Example Page
coco98.hasura-app.io	My profile page
coco98.hasura-app.io/article-one	My first article
coco98.hasura-app.io/article-two	My second article
coco98.hasura-app.io/article-three	My third article

Introduction to Modern Application Development | Dr. Ganesh Rajan (BIT Madras), Tamasz Csepel (Hasura)

So, if you look at my web app which is coco98.hasura-app.io, the final target is to have three modes of URLs available slash article one, slash article two, and slash article three.

(Refer Slide Time: 00:56)



1. Add URLs for each webpage

Deploy the web pages on a server

Make nice URLs for each page

Introduction to Modern Application Development | Dr. Gaurav Ramesh (BIT Madras), Tamasz Csepel (Hamburg)

This slide features a title '1. Add URLs for each webpage' and two bullet points: 'Deploy the web pages on a server' and 'Make nice URLs for each page'. A small video inset of the speaker is visible in the top right corner. The footer contains the course name and the speaker's names.

This is how we will proceed. We will add URLs for each web page, and we will first test; if URLs are working, and we are able to serve some responses on them.

(Refer Slide Time: 01:09)



URL based routing on a web server

- Write a web server
- Based on the URL, that is requested, return some content

If you remember from module P1: Apache, nginx, http-server were pre-configured to take the input URL and convert them into a filepath. So this logic mapping URLs to specific files need not be written by us.

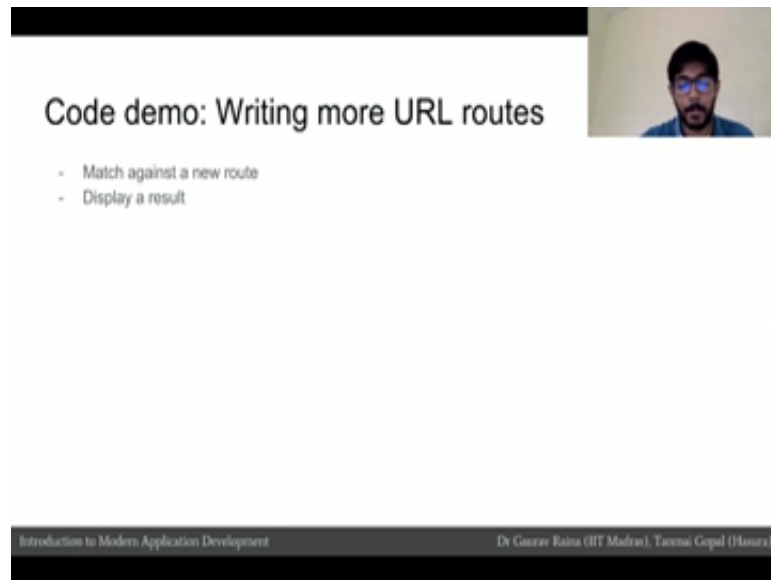
Introduction to Modern Application Development | Dr. Gaurav Ramesh (BIT Madras), Tamasz Csepel (Hamburg)

This slide features a title 'URL based routing on a web server' and two bullet points: 'Write a web server' and 'Based on the URL, that is requested, return some content'. A paragraph below explains that Apache, nginx, and http-server were pre-configured to handle URL to filepath mapping. A small video inset of the speaker is visible in the top right corner. The footer contains the course name and the speaker's names.

To do this, we have to understand how URLs based routing works. If you remember from module p1, when we are using http server and I mentioned that we could we used Apache and nginx as well, we notice that they were pre configured that they would take any input URL and convert that into a filepath. They would search for that file within the same directory that they were running in or they were configure to use and then they

would try to serve the file from that path if it is available. We are going to do something similar, but we are going to write the logic of what happens when you are going to particular URL.

(Refer Slide Time: 01:44)



Code demo: Writing more URL routes

- Match against a new route
- Display a result

Introduction to Modern Application Development Dr Gaurav Rana (IIT Madras), Tanmay Gupta (Amazon)

Let us see how that happens.

(Refer Slide Time: 01:52)



Chrome Home | All contents

Home | [Create](#) | [Forgot ID](#) | [Logout ID](#)

My settings
[Go to your user details](#)

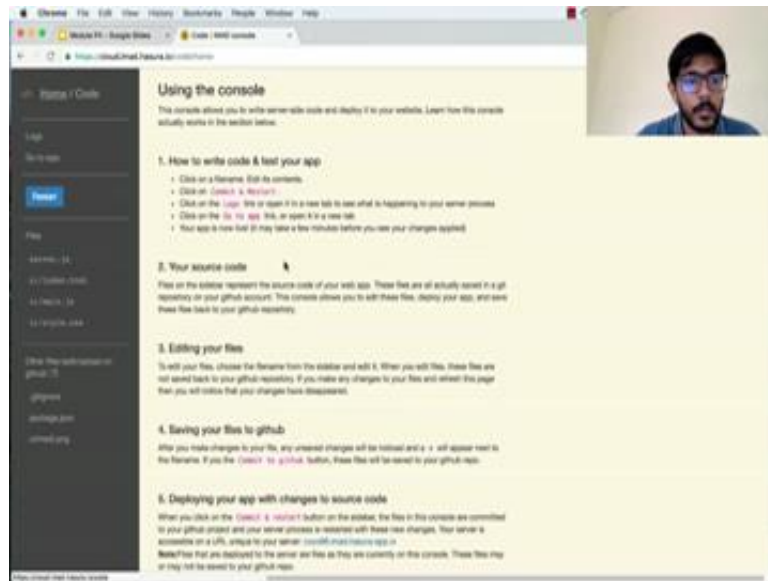
My database
[View Database Connections](#)

My server
username@hostname | local@bash local@aws-ec2-
password | ssh -oAdd -i /dev/null

[Go to your terminal](#) | [Send file through scp tool](#)

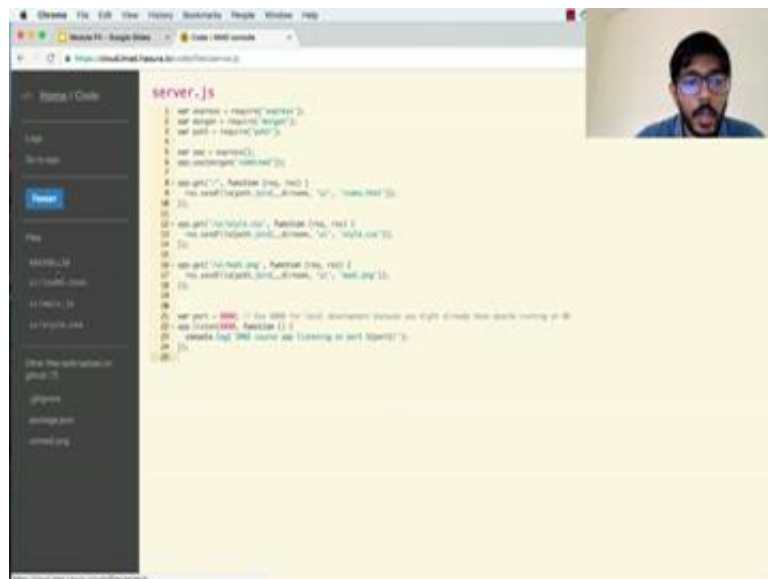
Introduction to Modern Application Development Made with ❤️ by Hemanth

(Refer Slide Time: 01:57)

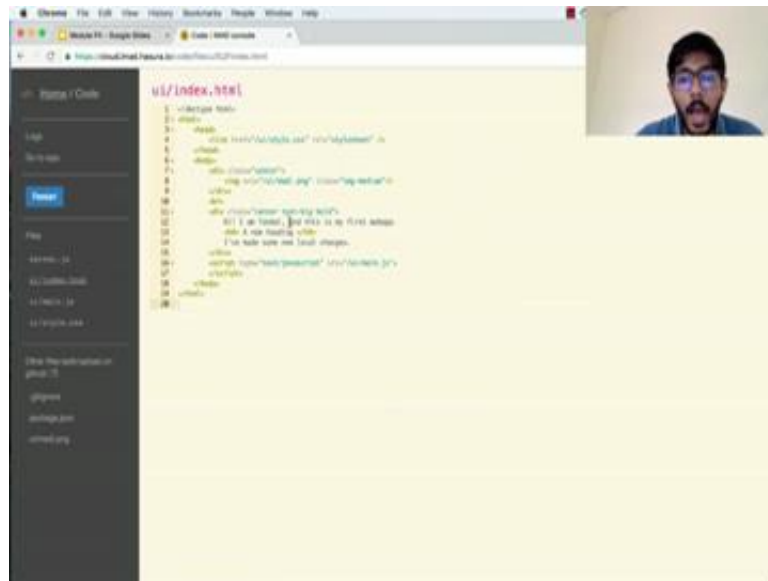


I first go to my console. I go to my code console

(Refer Slide Time: 02:00)



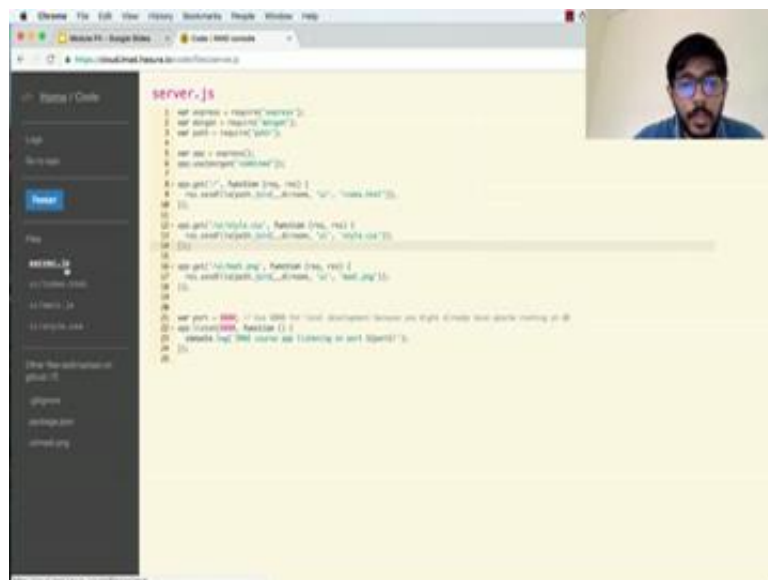
(Refer Slide Time: 02:01)



```
us/index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>us/index.html</title>
5 </head>
6 <body>
7 <h1>us/index.html</h1>
8 <p>us/index.html</p>
9 </body>
10 </html>
```

Now this is where we left off last time, where I made some changes on the server side to my HTML file, and then I made some changes from a local computer and then I deployed this site, this is where we left off last time.

(Refer Slide Time: 02:13)



```
server.js
1 const express = require('express');
2 const app = express();
3 const port = process.env.PORT || 3000;
4 const bodyParser = require('body-parser');
5 app.use(bodyParser.json());
6 app.use(bodyParser.urlencoded({ extended: true }));
7 app.get('/', function (req, res) {
8   res.send('Hello World');
9 });
10 app.get('/api/hello', function (req, res) {
11   res.send('Hello World');
12 });
13 app.get('/api/hi', function (req, res) {
14   res.send('Hi');
15 });
16 app.get('/api/bye', function (req, res) {
17   res.send('Bye');
18 });
19 app.listen(port, () => {
20   console.log(`Example app listening on port ${port}`);
21 });
```

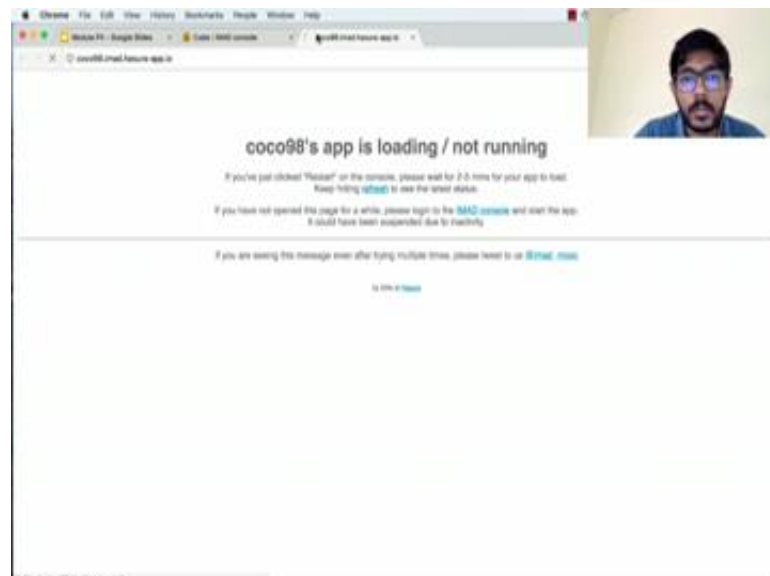
The first thing that we want to is add three URLs and search test responses from this URLs. Head to your server dot js 5, your server dot js 5 is the file that is actually executed and it is the source code of your web server. Let us quickly try to read the source code of this web server to understand what is happening.

In the first three lines, we were importing certain software packages. These are very commonly used libraries express is the library that is used for us to create the web server, so that we do not have to do the work of learning how to listen on a port or handling a stiff connections. We use the library Morgan to help us output logs of a servers that we know what request are coming to a server, and how we are responding.

The next few lines - line number 8, 12, and 16 are lines of code that are handling specific URLs. So, for example, here we have app dot get slash, and when a get request is made to slash, this function should execute. In on line number 12, we have app dot get slash u y slash file dot c s s, which says that if this URLs is requested a get request is made to this URL this function is executed.

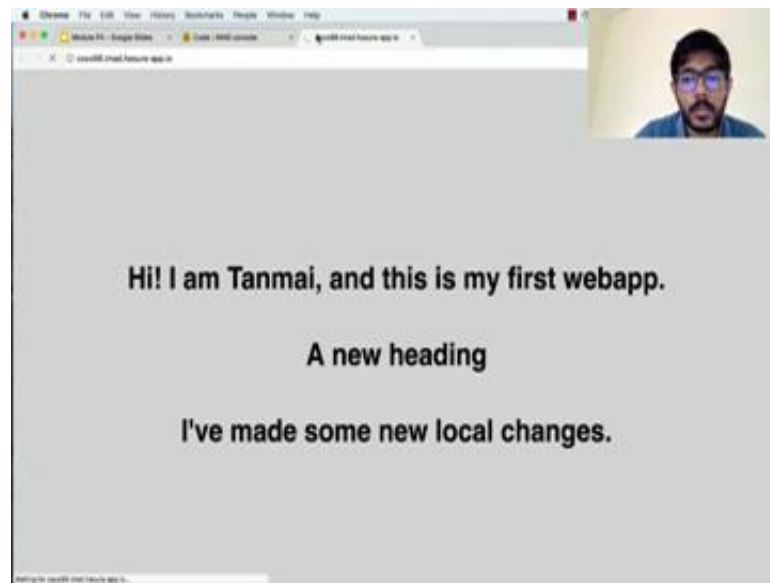
And similarly on line 16, if this particular URL path is requested, this function will execute. So, what would these functions do the first one if the browser or a client makes a request to the URL path slash, we use the send file function to pick up the file UI slash index dot HTML which is available to us. And we send the content of that file. Similarly, when UI slash dot style dot CSS is requested, we send of that file; and when the images requested we send of the image. Let us see how this works in action.

(Refer Slide Time: 04:13)

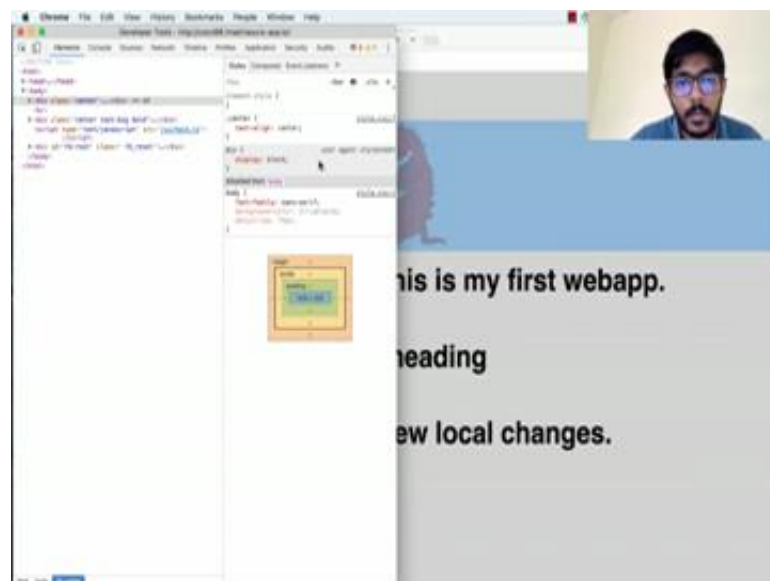


So, I restart my server, we will wait for the server to start.

(Refer Slide Time: 04:14)

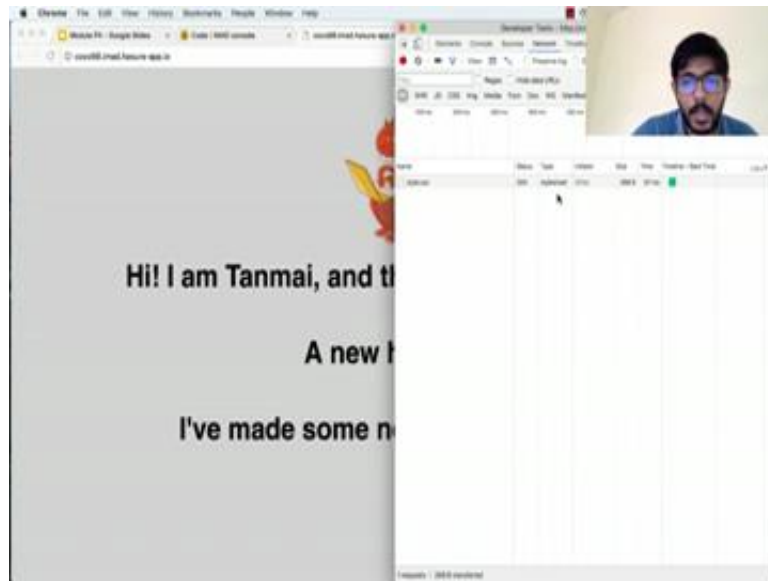


(Refer Slide Time: 04:19)



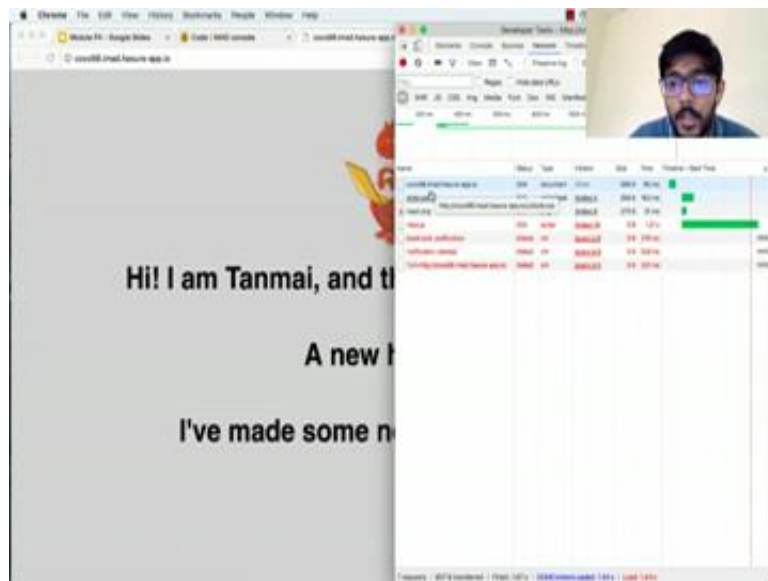
So, if I click on inspect element, like we had last time.

(Refer Slide Time: 04:23)



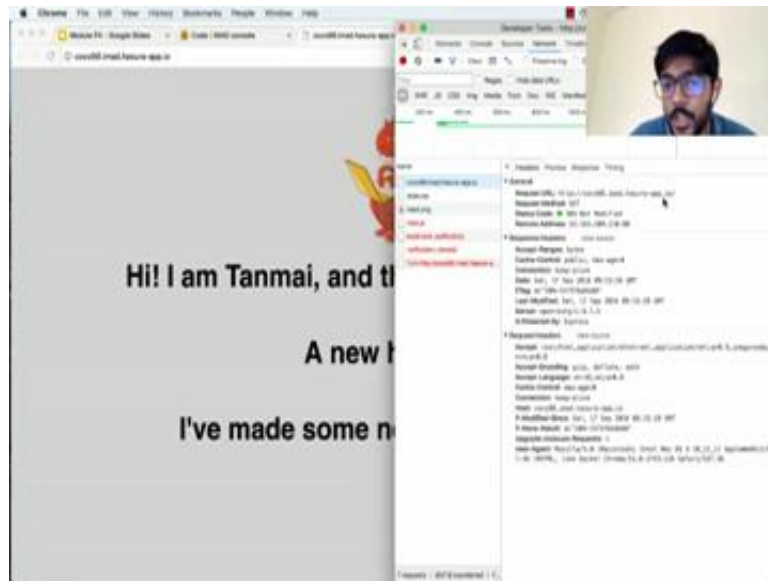
And I had to the network tab.

(Refer Slide Time: 04:25)



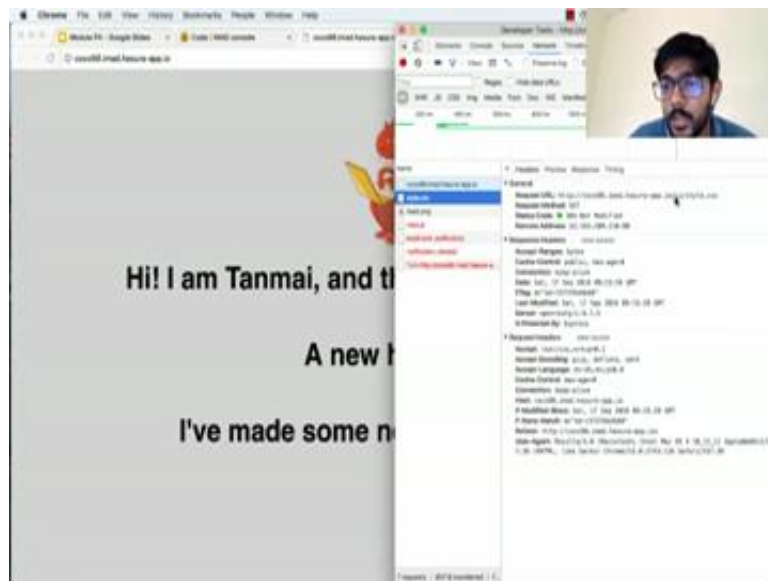
And I refresh the page again. You will see that three requests are made; and if you look at these three requests.

(Refer Slide Time: 04:33)



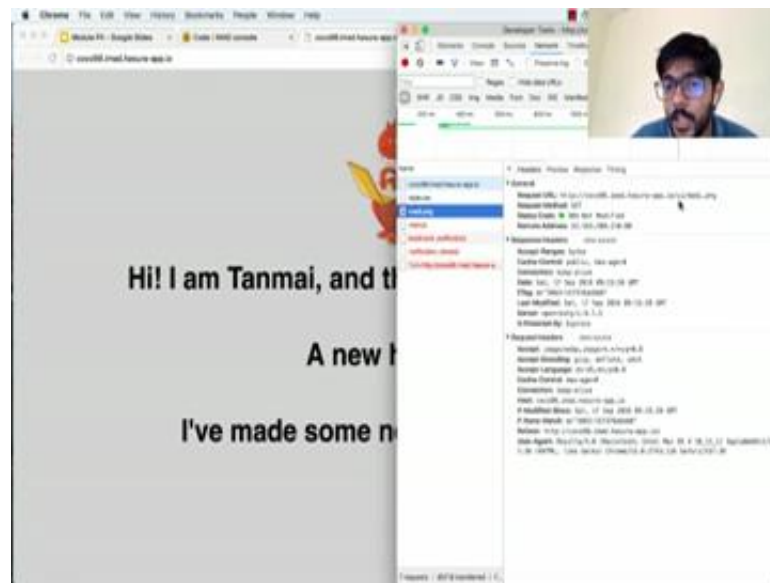
The first request is the first request is to our domain slash

(Refer Slide Time: 04:40)



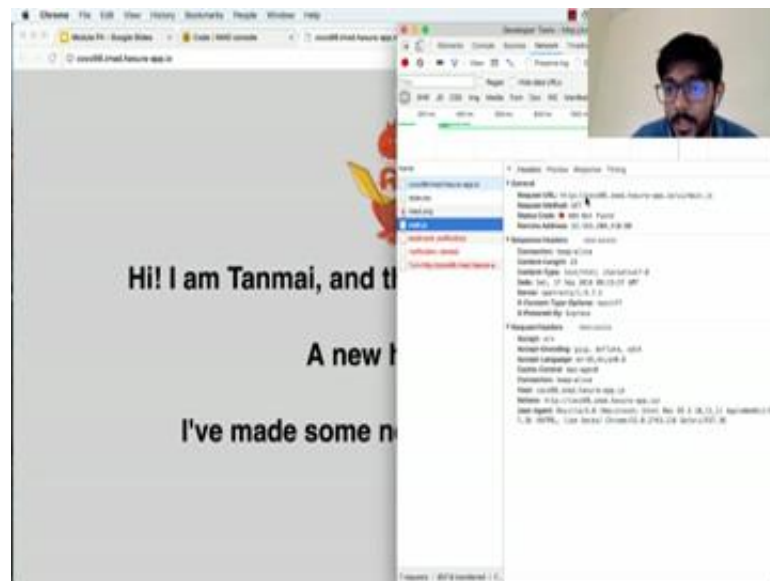
The second request is to slash UI slash dot file dot CSS.

(Refer Slide Time: 04:44)



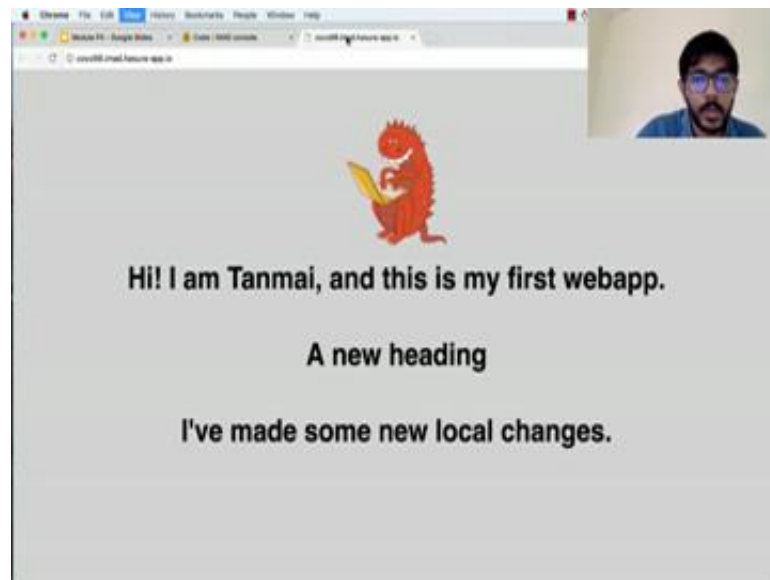
And the third request is to slash ui slash (Refer Time: 04:45) png.

(Refer Slide Time: 04:48)



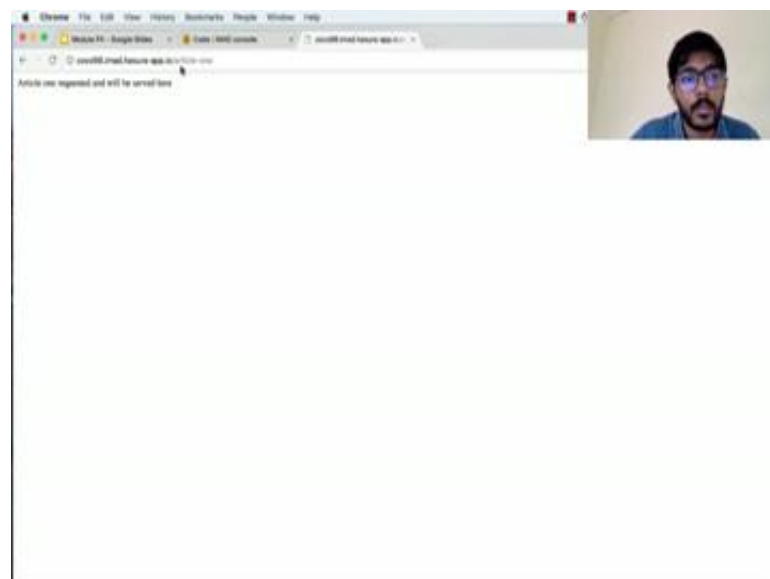
In fact, it is also making a request to a java script file slash ui slash main dot js, but we do not have any server side code that handles this URL, which is why the server is responding with the 404, which means that this URL path was not found or not handled by a web server. We will correct this part of a web server, when we come to client side java script in the next few modules.

(Refer Slide Time: 06:49)



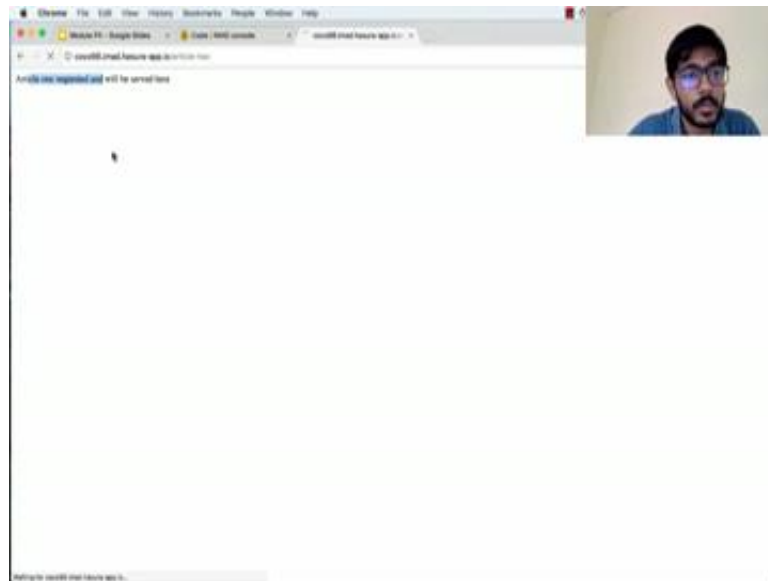
Let us go and refresh this page unless had to slash article 1.

(Refer Slide Time: 07:00)



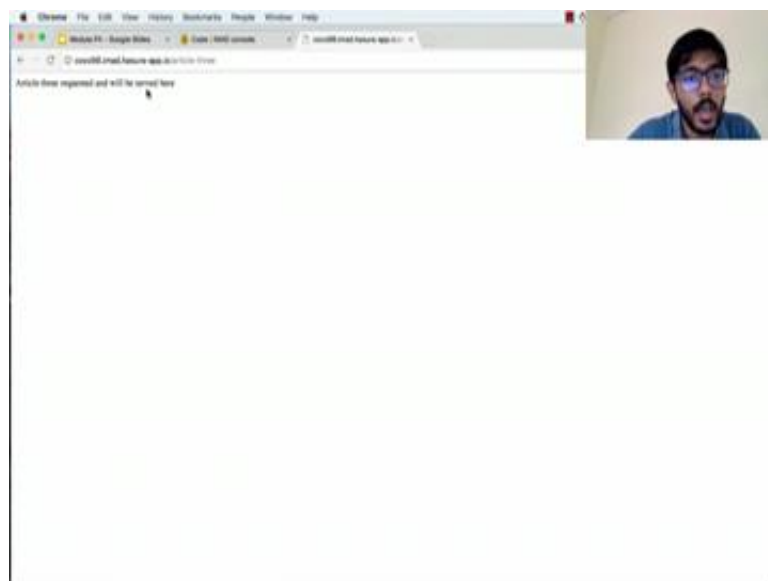
And you can see that the article 1, we are getting a response here.

(Refer Slide Time: 07:03)



Let us check out article 2, and you seeing article 2.

(Refer Slide Time: 07:10)



Let us switch this out to article 3, and you seeing article 3. So, all three URLs seem to be working great.

(Refer Slide Time: 07:16)



2. Make simple web-pages

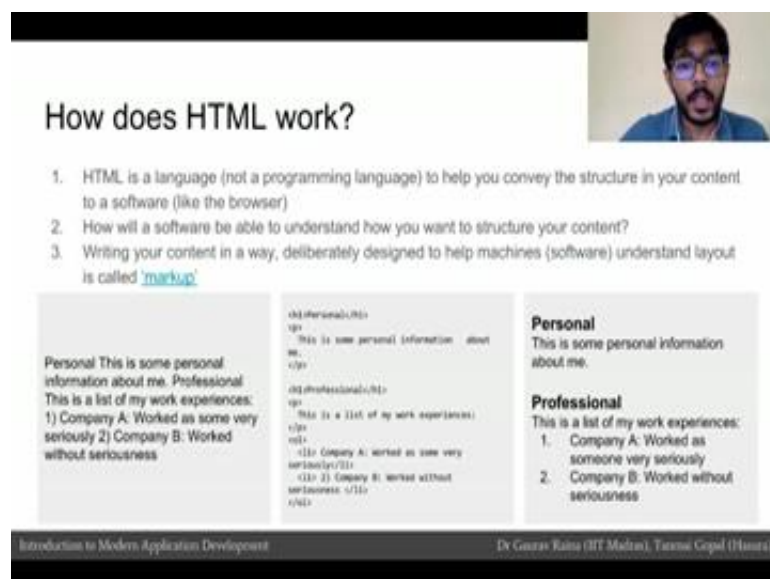
Need to make content and make it look good!

Hence, HTML + CSS

Introduction to Modern Application Development | Dr. Gaurav Rainsa (BIT Madras), Taranjeet Gopal (Hassart)

So, now that we have our URLs and we are able to serve responses on those URLs. We need to make simple web pages and then ultimately, we will serve these web pages from on URLs. So, how do we make a web page? The language at the browser understands to be able to show your web page is HTML and CSS. The browser can understand these two languages HTML and CSS, and render them to create web pages the way we use them on the internet. Let us understand this in a little more detail.

(Refer Slide Time: 07:46)



How does HTML work?

1. HTML is a language (not a programming language) to help you convey the structure in your content to a software (like the browser)
2. How will a software be able to understand how you want to structure your content?
3. Writing your content in a way, deliberately designed to help machines (software) understand layout is called **'markup'**

<p>Personal This is some personal information about me. Professional This is a list of my work experiences: 1) Company A: Worked as some very seriously 2) Company B: Worked without seriousness</p>	<pre><div personal/> <p> This is some personal information about me. </p> </div> <div professional/> <p> This is a list of my work experiences: </p> 1) Company A: worked as some very seriously 2) Company B: worked without seriousness. </div></pre>	<p>Personal This is some personal information about me.</p> <p>Professional This is a list of my work experiences:</p> <ol style="list-style-type: none">1. Company A: Worked as someone very seriously2. Company B: Worked without seriousness
--	---	--

Introduction to Modern Application Development | Dr. Gaurav Rainsa (BIT Madras), Taranjeet Gopal (Hassart)

HTML is a language; it is not really a programming language. And the purpose of HTML is to convey the structure that content has; it is means for like any language, like any human to computer language, it is a language that is meant to be written by humans, and to be understood by software. Whenever a language is understood by software, the language will have what is called is syntax, it will have a very particular and a very definite structure. And this structure is something that can be automatically understood by the machine or by the browser, and converted into a visual display.

Let us look at an example how HTML works. On the left most side on this box, you see that there is some content written; this is content that would typically go on to profile page. And you can notice this content is just text; this is not the way we would like to display content on the profile page. We would like to display it like how we have on the right here. So, how do we, how do we help the browser understand that this text has to be displayed in this way right. To do this, we use the language HTML.

And so let us look at what the HTML looks like, the first thing that we do is we say `h 1` personal slash `h 1`. So, this part is used to represent that `h 1` is starting, and so it is a star heading the content of the heading which is text which is personal. And then we say slash `h 1` which is meant to say end the heading.

Similarly, we start a paragraph we put the content here, and we end the paragraph. Then we start a new heading, professional and we end the heading; we start a paragraph, we end the paragraph. Then we start order list - `ol` is intend to represent order list. So, we start an order list, and we end the order list here. The first element in that list is your first professional experience; the second element of that list is the second experience. So, this is how we moved from text to HTML and the browser understood the HTML and can displayed like this.

(Refer Slide Time: 09:47)

Add style to structure

1. But structuring is not styling
2. If something is a heading, how should it be styled? What is the visual meaning of that structure?
3. What does the browser do?
 - a. Lots of defaults ;)
 - b. You can customize defaults and add more styling & formatting using CSS

<pre><h1>Hello World!</h1> <p> This is some personal information about me. </p> <h2>Professional</h2> <p> This is a list of my work experiences: </p> Company A: Worked as some very serious Company B: Worked without seriousness </pre>	<p>Personal This is some personal information about me.</p> <p>Professional This is a list of my work experiences:</p> <ol style="list-style-type: none">1. Company A: Worked as someone very seriously2. Company B: Worked without seriousness	<pre>h1 { text-decoration: underline; } p { color: grey; } body { background: white; }</pre>
--	--	--

Introduction to Modern Application Development | Dr. Gaurav Raina (IIT Madras), Tanmay Gupta (IIT Madras)

But if you think about it, we actually did not tell the browser how to make the content look we help the browser understand what the sections of vertex star. So, we said that this text has many sections, the first section is the heading, the next section is a paragraph, then there is another heading then there is a paragraph and there is an order list. But how does the browser know what an order list is, how does the browser know that you know this heading should be displayed in a larger font size, it should be bold, it should have more spacing or how does the browser know that there is a list that should be a little bit of space on the left. And then there should be I there should be one and then there should be another pieces space here, how does the browser know this.

And we call this styling right, this is not structure; this is style. And the browser understands style using a language called CSS, but even if you do not specify a custom style for what a heading is, all browsers have lot of defaults. And the browser renders and the browser renders particular structure elements with some default styling. So, for example, by default, the browser will represent a heading in this particular way. The browser will represent the heading paragraph in this particular way, lists in this particular way.

Suppose we want to customize these default strike, you would you would write in a language called CSS and in this CSS language you will specify what the styling for the structure element is. And so, here we have a (Refer Time: 11:15) CSS, and the CSS

language which is very different from the HTML language; it says that for the h 1 element apply this styling, for the p element apply this styling, for the body element apply this styling. Let us see where they look like.

(Refer Slide Time: 11:29)

The slide, titled "Add style to structure", illustrates the process of applying CSS to HTML. It is divided into three main sections:

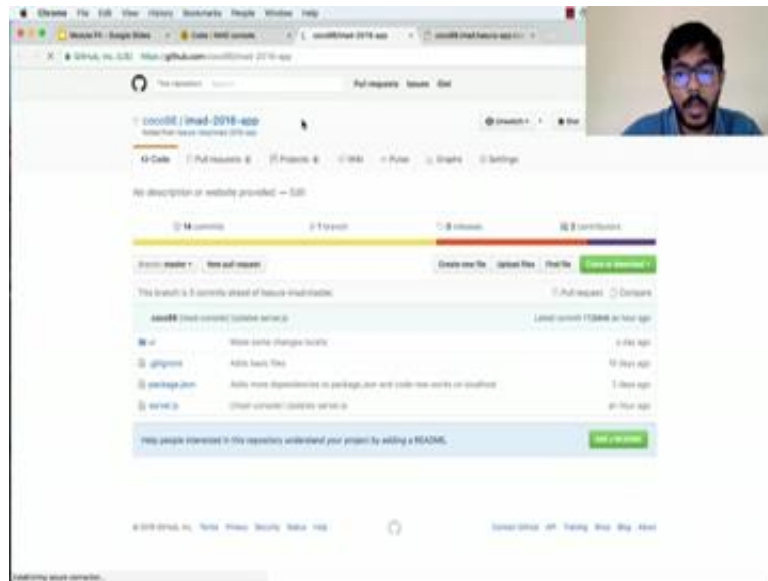
- HTML Code:** Shows the structure of the document with two sections: `<div personal />` containing a paragraph about personal information, and `<div professional />` containing a list of work experiences. A red arrow points from the `<h1>` tag in the professional section to the visual rendering.
- Visual Rendering:** Shows the browser's default rendering of the HTML. The `<h1>` is bold and larger, the `<p>` is gray, and the `<body>` is white. A green arrow labeled "CSS" points from the CSS code block to this rendering.
- CSS Code:** Shows the CSS rules being applied: `h1 { text-decoration: underline; }`, `p { color: grey; }`, and `body { background: white; }`. A green arrow points from this code to the visual rendering.

The slide also features a small video inset of a speaker in the top right corner and a footer with the text "Introduction to Modern Application Development" and "Dr. Geetanjali Kulkarni (BIT Mesra), Tejaswi Gupta (IITM)".

By default the browser was converting this HTML into this visual rendering, but if we apply this CSS where we have seeing h 1 should have a text decoration underline. The p paragraph should have a font color gray and the entire body which is everything that is around the HTML should be a background white.

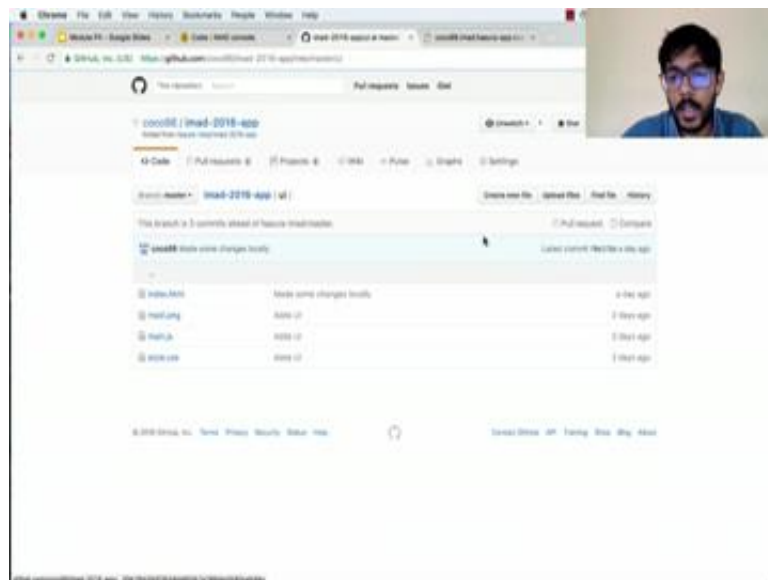
And so we if we apply this CSS the browser will not render your HTML like this, but it will render the HTML like this, and here you can see that more style is being applied the heading is underlined. It is still has the default styling which is bold and larger font size, but it also has an underline. So, it our CSS added to the default CSS the browser had. Similarly, the paragraph as colored gray, but the original defaults of the browser in the way then does the paragraph goes preserved and we did not modify the way the list is render.

(Refer Slide Time: 12:37)



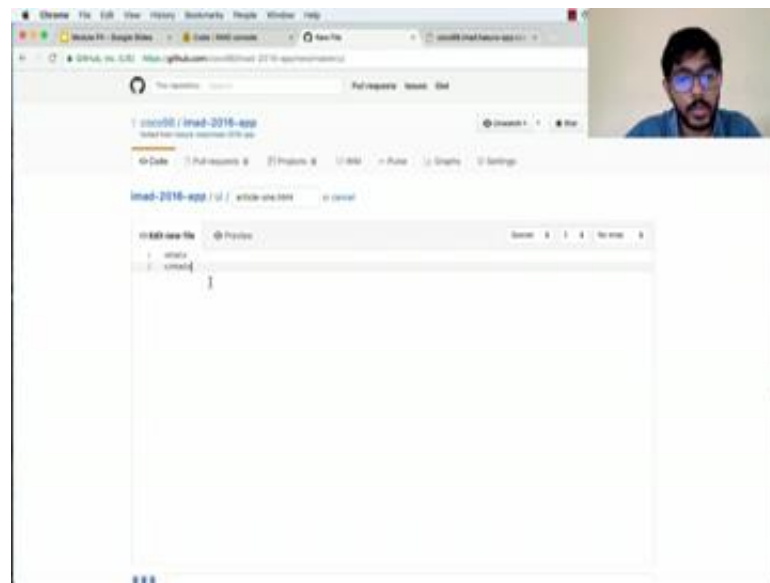
I have to add a new page. So, I can add a new page by directly going onto github. So, go to data github.

(Refer Slide Time: 12:39)



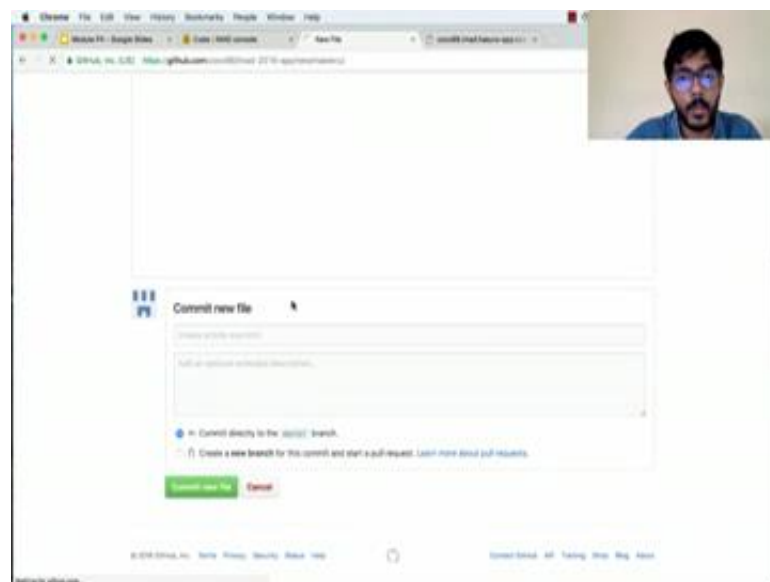
I go to the UI folder.

(Refer Slide Time: 12:41)



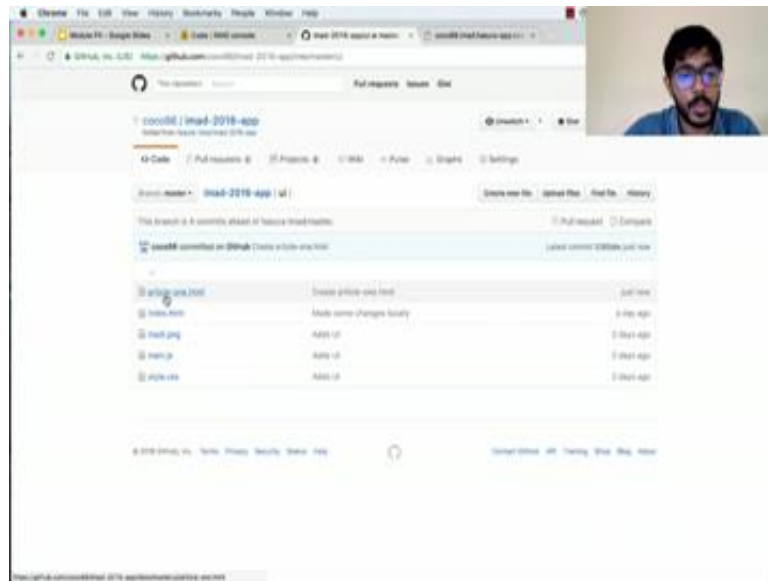
I say create new file and I will name my file article one dot HTML, and I m just going to put in some data because we will actually edit these files in a consoles that is a HTML.

(Refer Slide Time: 12:59)



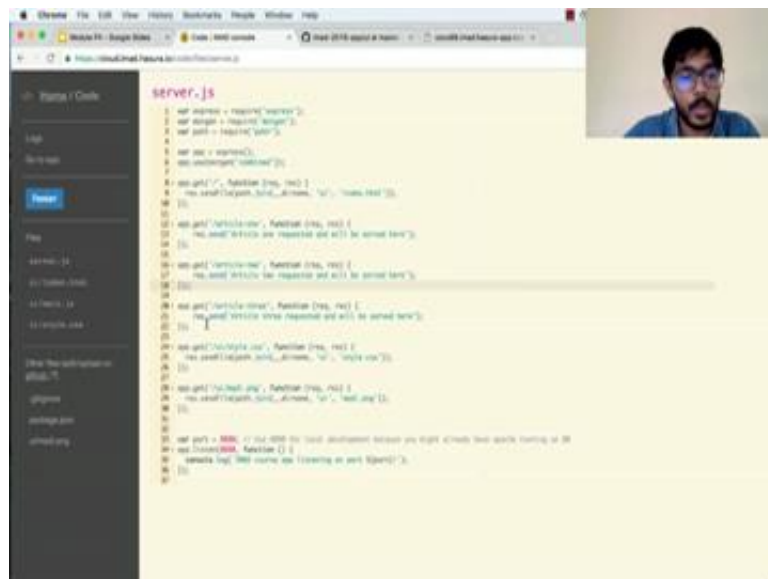
There is a commit new file.

(Refer Slide Time: 13:00)



Let saves the file here.

(Refer Slide Time: 13:03)



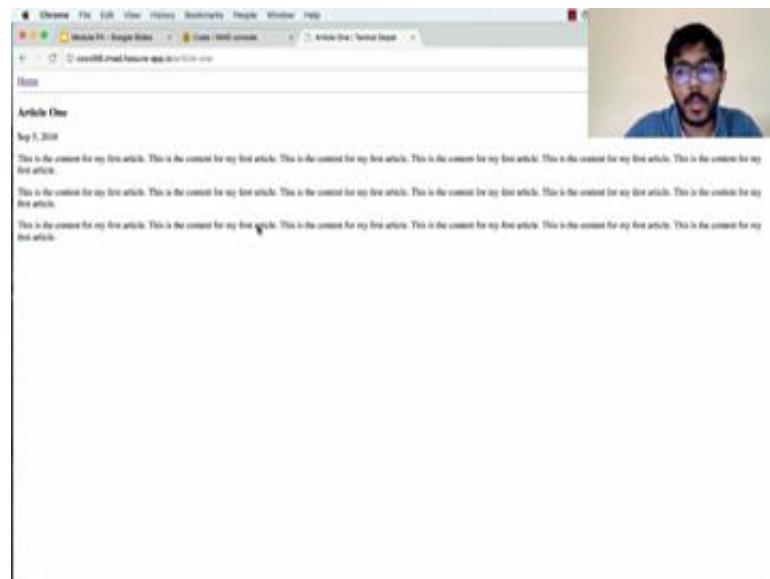
Let us go back to our console.

(Refer Slide Time: 16:15)



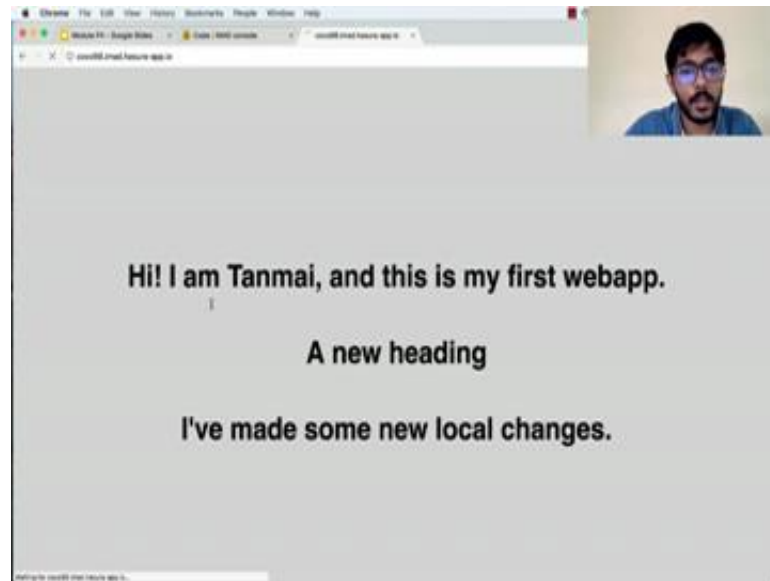
And see; wait for our server to start.

(Refer Slide Time: 16:17)



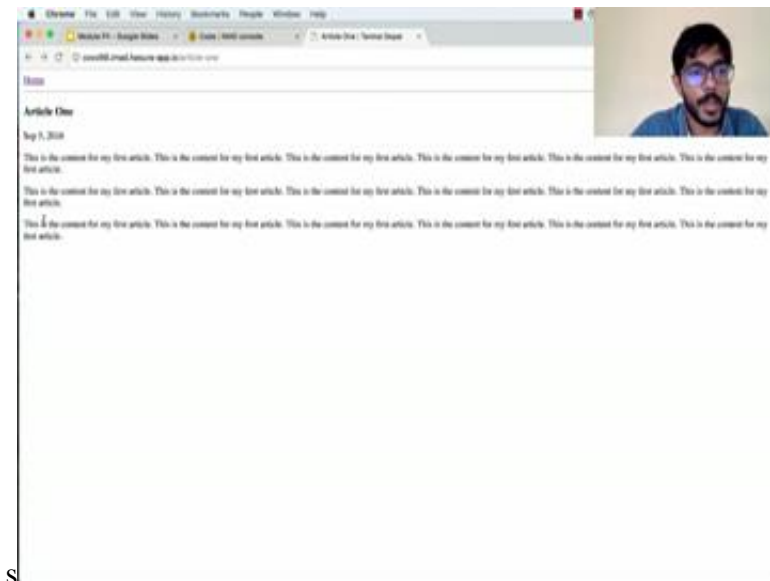
And there we go. So, we have our first HTML page that is ready.

(Refer Slide Time: 16:24)



You can see that right click on the home link.

(Refer Slide Time: 16:25)



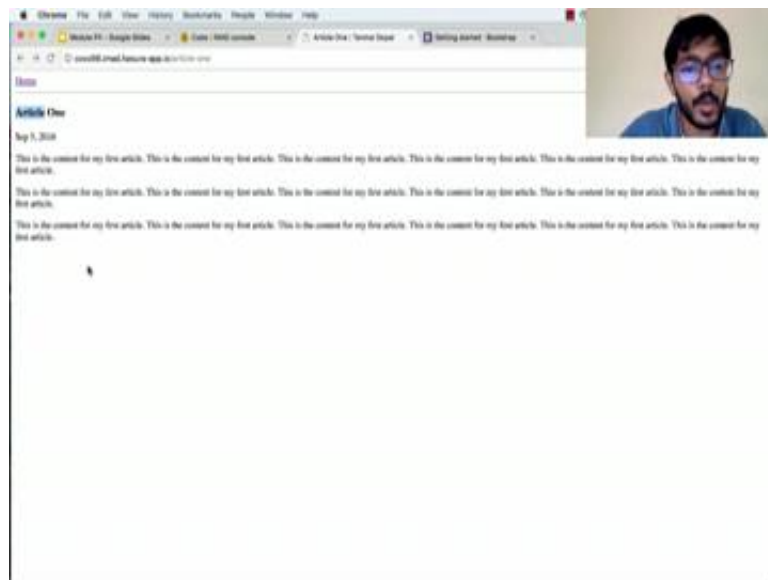
I will go back to the home page. And otherwise, I have may add to content you can see this is the heading there is little bit of a space here and then there are paragraphs. And whenever I create a paragraph element, the default styling of the browser is automatically is updating that paragraph space.

(Refer Slide Time: 16:43)



What we also do it to our HTML file is that we will add a special tag. So, we will add this special tag and this tag is for mobile browsers.

(Refer Slide Time: 16:56)



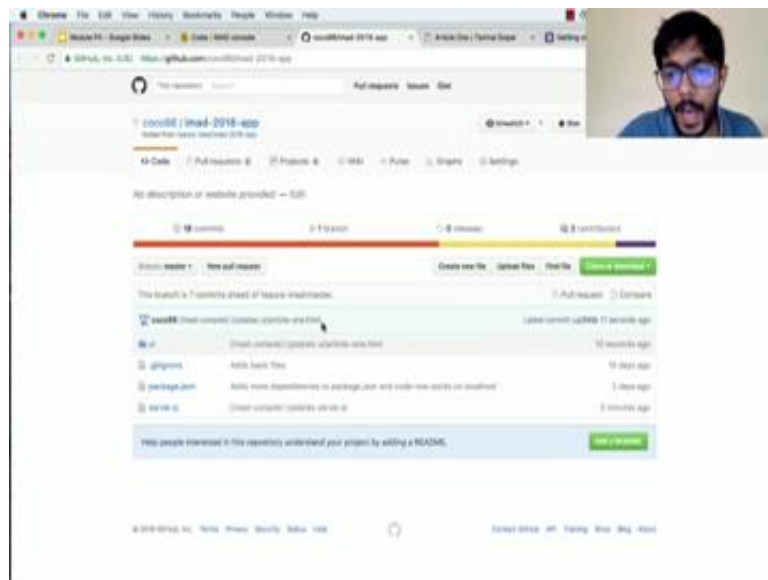
So, that mobile browsers do not display a website in a way that is zoomed out and you do not have to zoom in to give this website on mobile list will automatically work on mobile browsers, and we will understand how will works on later on.

(Refer Slide Time: 17:04)



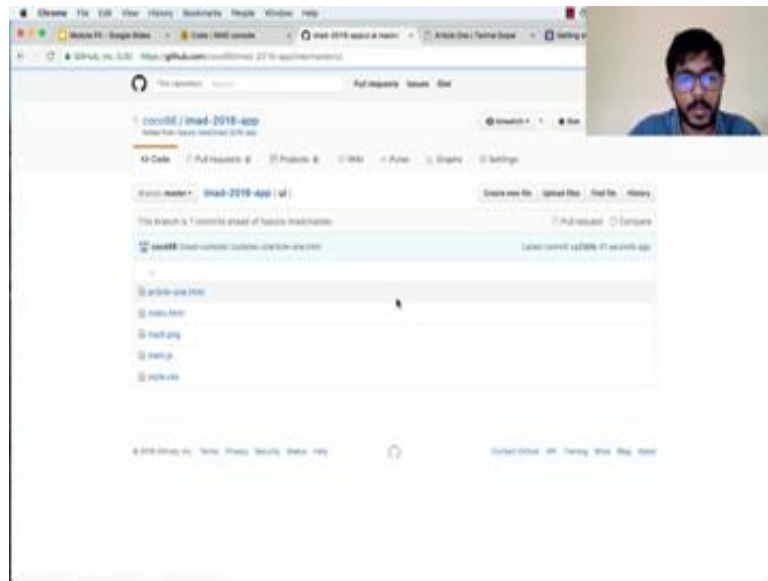
So, this is my HTML file, I am going to commits this again, so that is saved to github project. Now let us go ahead and add some more files.

(Refer Slide Time: 17:20)



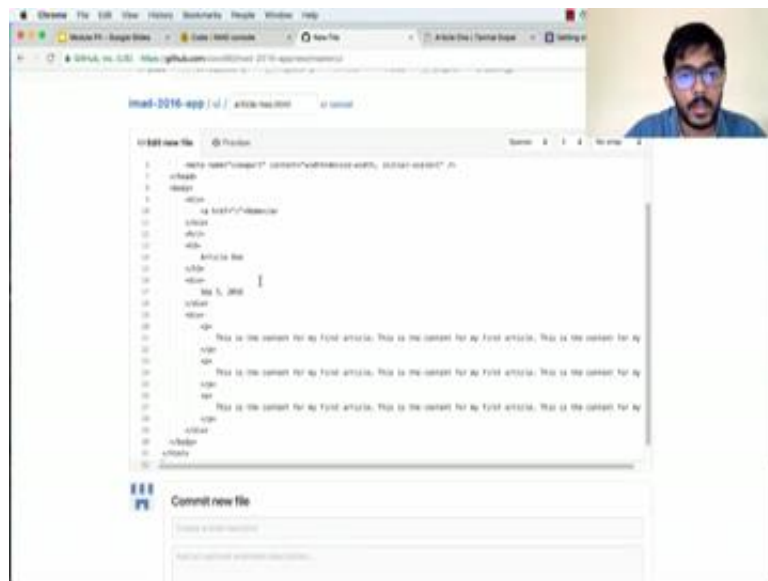
So, I go back to github. Let us quickly add some more files.

(Refer Slide Time: 17:22)



I go back to the UI folder.

(Refer Slide Time: 17:24)



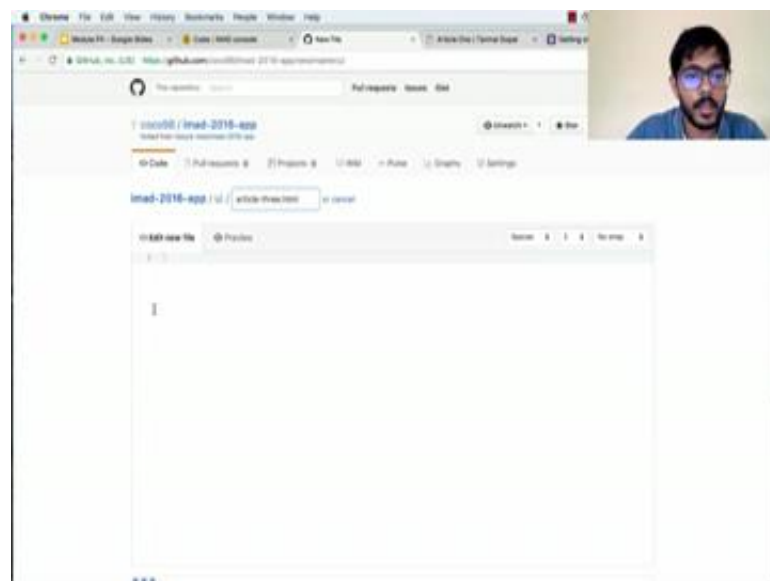
I am going to say create new file article-two. html. So, here we have article-two.html. And I am just going to copy the content from my original article, paste it here. I will change the content article-two. Let this article-two, which is the content for my second article. Let us save this file.

(Refer Slide Time: 17:52)



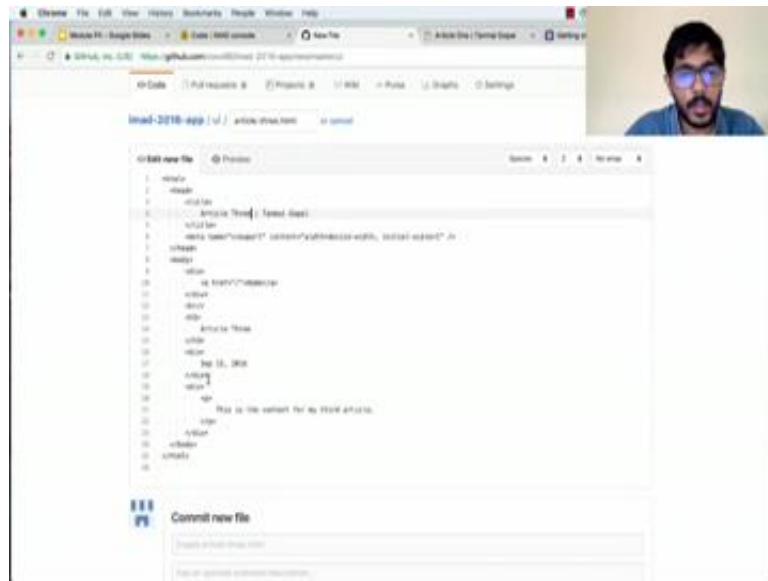
And let us quickly create a new file.

(Refer Slide Time: 17:56)



Let say article-three.html.

(Refer Slide Time: 18:09)

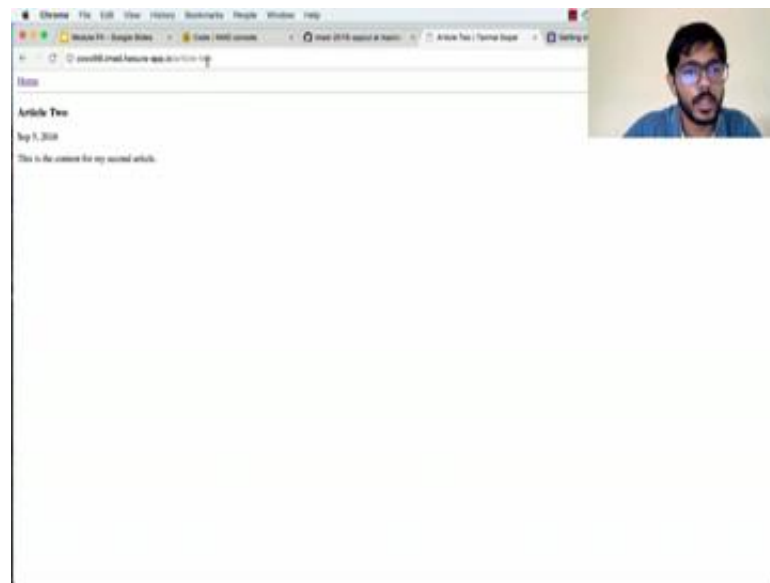


Copy, paste this data; and which this is my third article, I change the date here, and there we go, I just got my third HTML page.

(Refer Slide Time: 18:27)



(Refer Slide Time: 19:01)



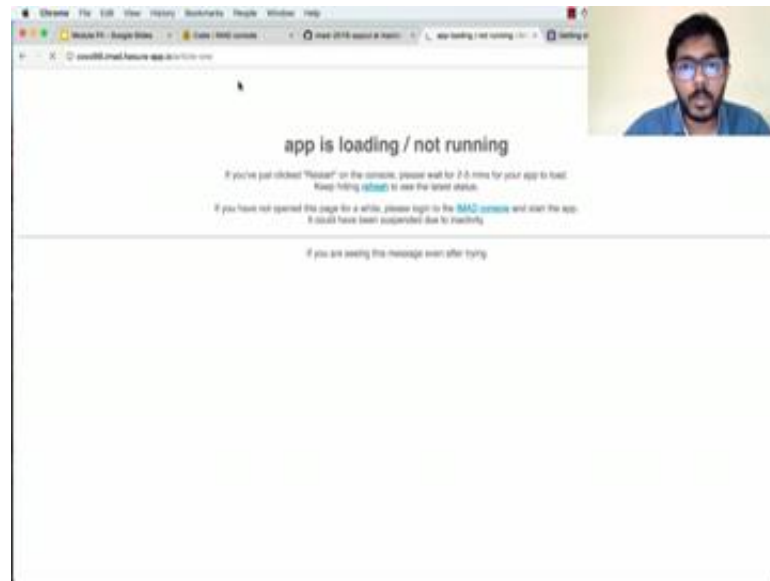
Let us head to article-two. And check if the things are working. Wait a little bit for a server to restart. And there we go there, we have article-two.

(Refer Slide Time: 19:11)



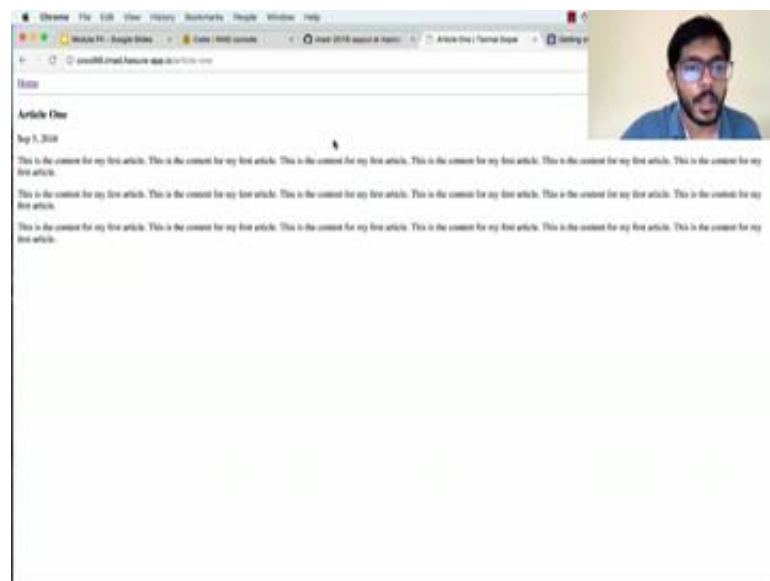
Let us go to article three and we have article-three contact method article, you can see the title here.

(Refer Slide Time: 19:33)



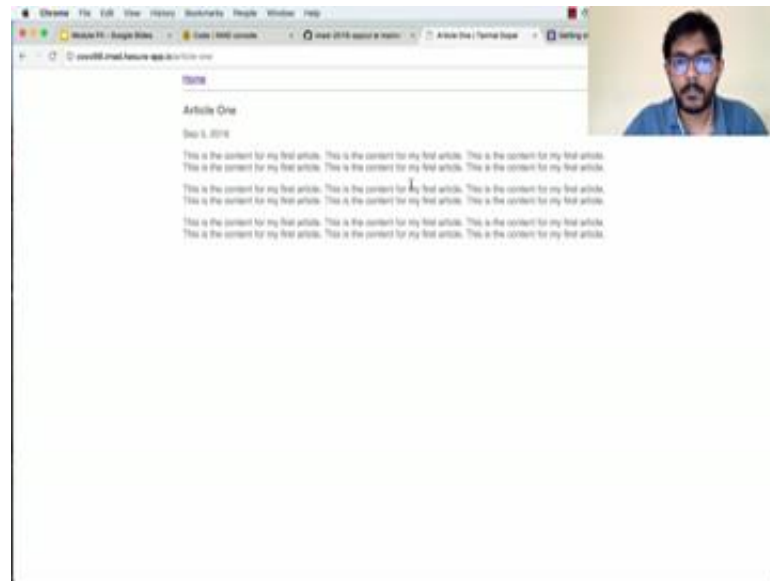
So, what I would like to do on my page.

(Refer Slide Time: 19:35)



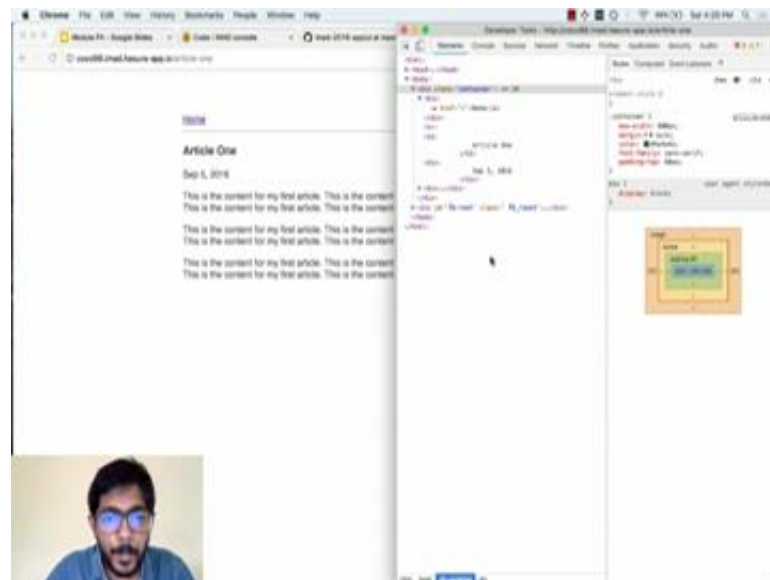
And let us look at the page article one. So, here we have article one. Now what I would like to do to make this page look better and I going to center the content little bit, this is too wide. I am also going to change the colors a little bit and change the font also.

(Refer Slide Time: 21:15)



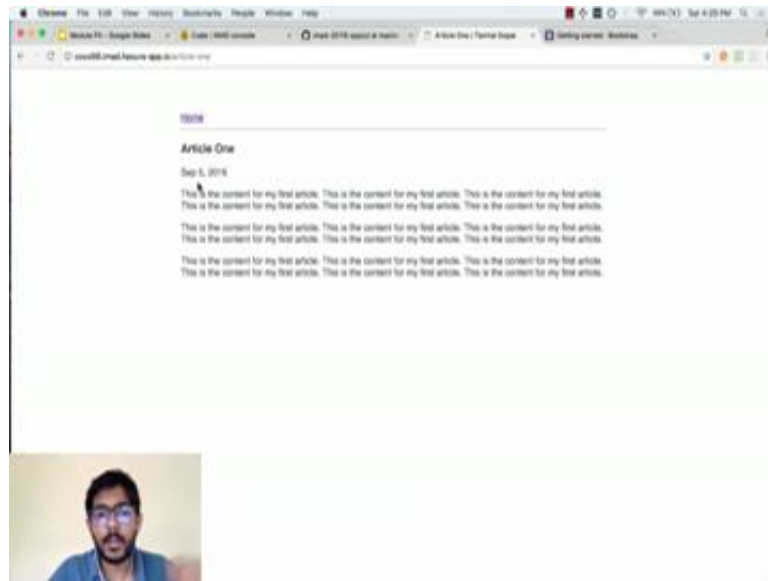
So, I restart my server and there we go. So, I have this it is already looking a little neater, because I can read my content more easily.

(Refer Slide Time: 21:25)



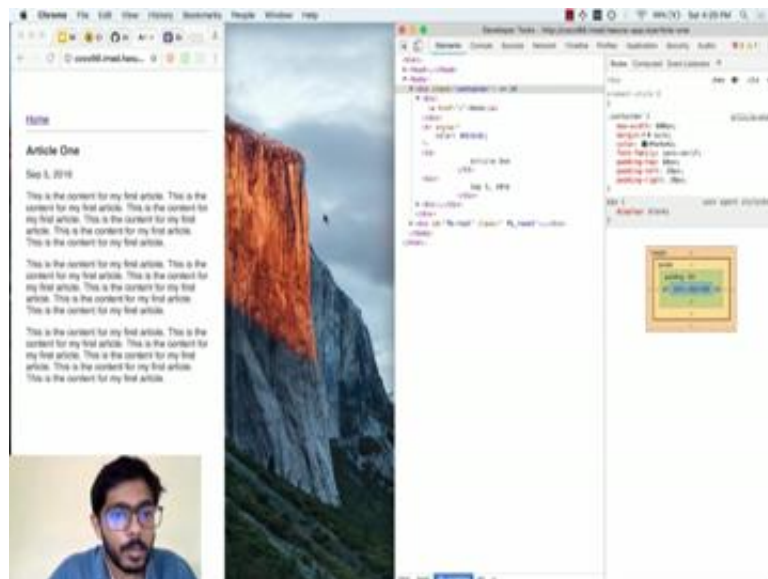
So, I want to figure out how to make this neater. So, I will first make these changes inside my inspect elements itself. Let us look at container and I think I should add a little bit of a gap on the top. So, may be 60 pixels for gap. The color is little too dark, so let us too lights, let us find a slightly better color, there we go and all right.

(Refer Slide Time: 21:56)



So, that makes my page look a little neater.

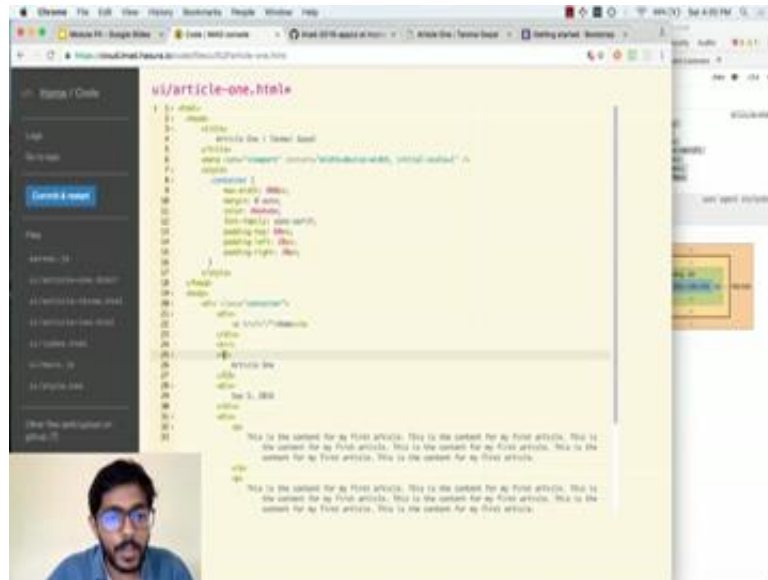
(Refer Slide Time: 22:10)



And let us just a resize a browser to see what it would look like on mobile. So, on mobile there is a little bit of a gap that is appearing that is not there on the slides. Let us go to our container and make sure that it has a little bit of padding on the left and the right hand side. So, I am going to say padding left of about 20 pixels, and padding right of about 20 pixels. This makes it a little easier to observe or look at content on mobile on the mobile browser.

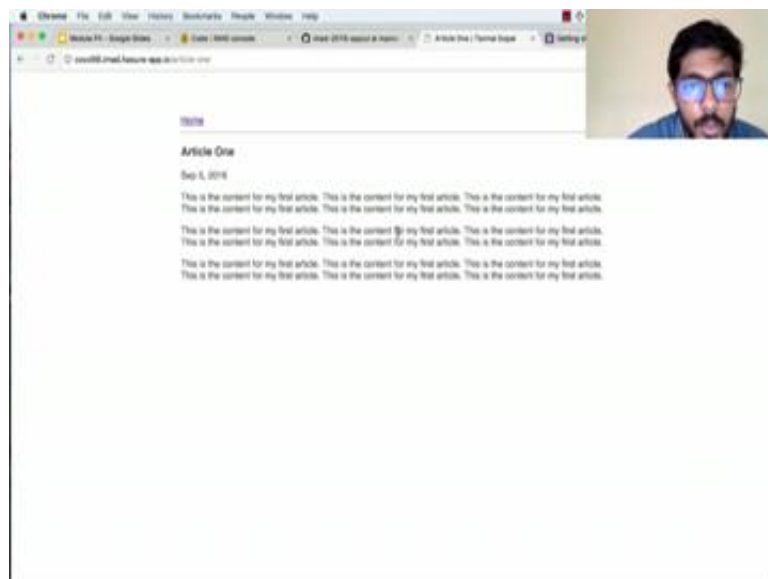
And let us resize this back to see; if everything is good, things are looking fine. So, I am going to take this CSS that I had in inspected element. And I am going to take this and seamlessly copy paste it back into my code.

(Refer Slide Time: 22:49)



So, there we go. It is align this to make it look a little better, there is nothing uglier to program with an (Refer Time: 23:01) code, cool.

(Refer Slide Time: 23:12)



So, I have done this, let us commit and restart. Wait for it to restart. And let us start a server again. And we have article-one that is working. So, now, I will have to go apply these changes on the other articles as well.

(Refer Slide Time: 23:17)



```
1 <!-- article-two.html -->
2 <html>
3 <head>
4 <title>Article Two</title>
5 </head>
6 <body>
7 <div class="container">
8 <h2>Article Two</h2>
9 <p>This is the content for my second article</p>
10 </div>
11 </body>
12 </html>
```

So, I will have to go to article two and copy paste these changes. And I have to go article three and also copy paste them. And once I do that I will have the same styling that is apply to article one, two and three. So, when we are copying this styling, we will have to also modify the HTML to add this sort of container elements, so that we can style a container element.

(Refer Slide Time: 23:42)



3. Write less code

Stop repeating the same styling information on each page

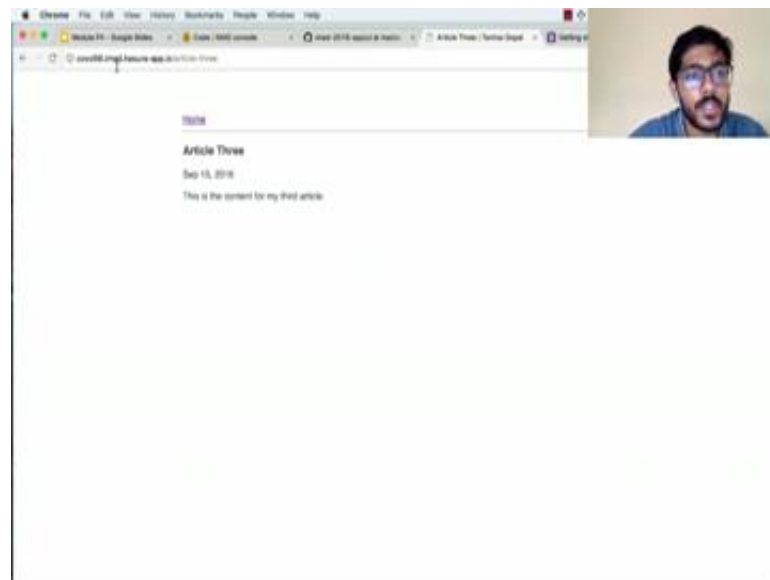
Stop repeating the same structure on each page

Quick Exercise:
Why is repeating code a bad thing?
Why is it potentially harmful?

Introduction to Modern Application Development | Dr. Gaurav Rana (BIT Mesra), Tanmay Gopal (IITM)

Finally, at the end of this write, we have achieved our goal of having created 3 URLs and having 3 pages, and having a little bit of styling on them as well.

(Refer Slide Time: 23:59)



Chrome | File | Edit | View | History | Bookmarks | Desktop | Window | Help

Article Three | Sep 15, 2016 | This is the content for my third article

For example, we are going to have article-one or article-two and there article-three. And we have 3 pages in this style them.

(Refer Slide Time: 24:02)



3. Write less code

- Stop repeating the same styling information on each page
- Stop repeating the same structure on each page

Quick Exercise:
Why is repeating code a bad thing?
Why is it potentially harmful?

Introduction to Modern Application Development | Dr. Geetanshi Rastogi (IIT Madras), Tanvika Chopra (Harvard)

Now if you observed I was I was copy pasting throughout the session, I was copy pasting the HTML I was copy pasting lines and server dot js and I was even copy pasting this CSS information. So, how can we avoid writing so much called the goal of a good programmer; is always to write the least amount of code possible.

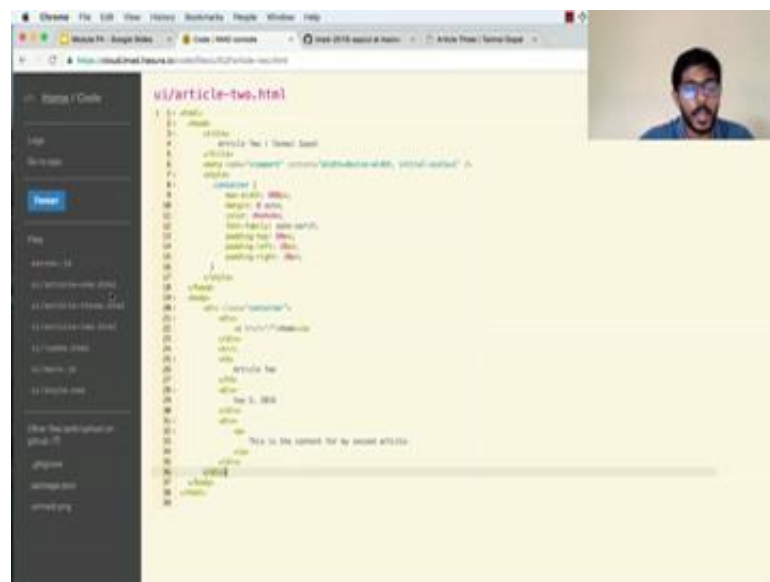
You should never repeat functionality by adding by repeating code; and apart from the aesthetically not very elegant, this is also potentially harmful. You guys should read up little bit on the internet about what is called the dry principle - the d r y principle; and why a repeating code here could be potentially harmful.

(Refer Slide Time: 24:40)



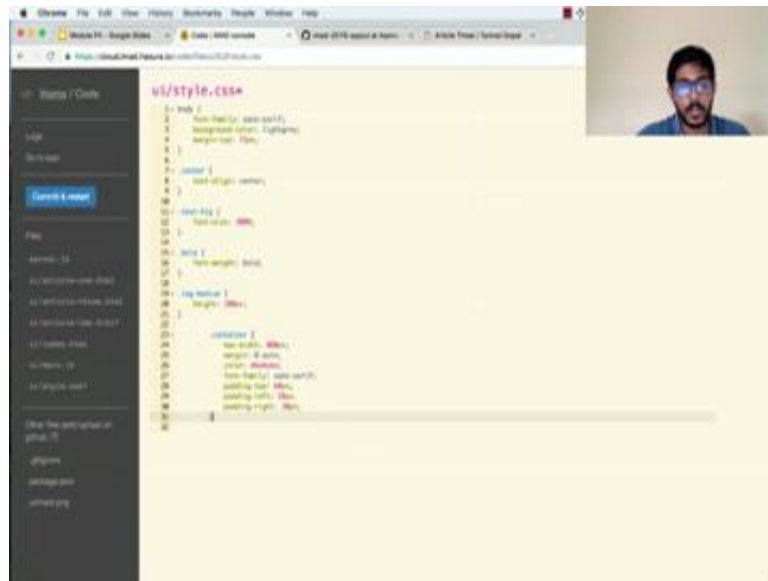
So, we have three specific areas where we you want to reduce the code written, common CSS, common HTML and common lines in server dot js. Let us see how we do this.

(Refer Slide Time: 24:51)



So, I go back to my code console. And I have already copied. So, I go back to my coding console. The first thing that I am going to do is removing this CSS, and moved it into separate files. So, I am going to take this CSS content.

(Refer Slide Time: 25:10)

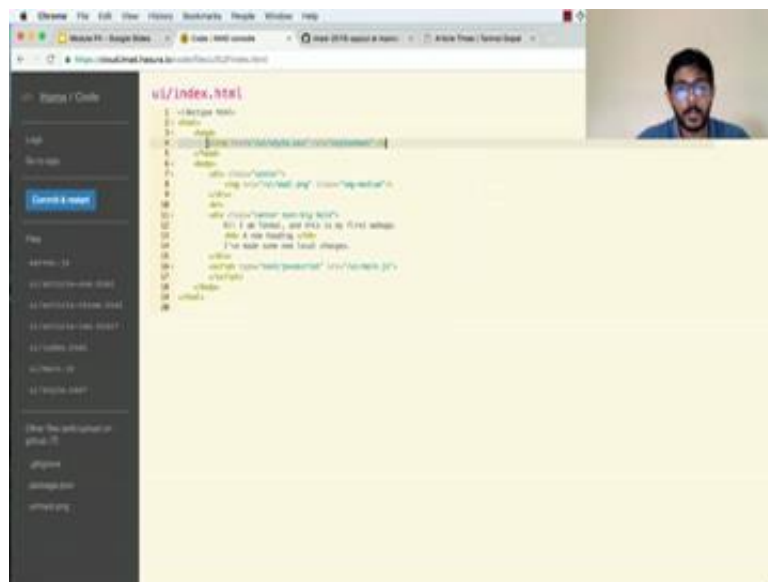


The screenshot shows a code editor with a dark sidebar on the left and a main workspace. The workspace displays the content of a file named 'ui/style.css'. The code is as follows:

```
1 body {  
2   font-family: sans-serif;  
3   background-color: #f4f4f4;  
4 }  
5  
6 .container {  
7   text-align: center;  
8 }  
9  
10 h1 {  
11   font-size: 2em;  
12 }  
13  
14 h2 {  
15   font-weight: bold;  
16 }  
17  
18 .my-button {  
19   height: 30px;  
20 }  
21  
22 .button {  
23   border: 1px solid #ccc;  
24   padding: 5px 10px;  
25   display: inline-block;  
26   margin: 10px 0;  
27   text-decoration: none;  
28   color: #000; background-color: #fff;  
29   border-radius: 4px;  
30 }  
31
```

And I am going to put it in style dot CSS. I am going to align this little bit. I will press shift tab to align that, and what I will do is now obviously, because I am using style dot CSS, I have to refer to this style file inside my HTML code.

(Refer Slide Time: 25:26)

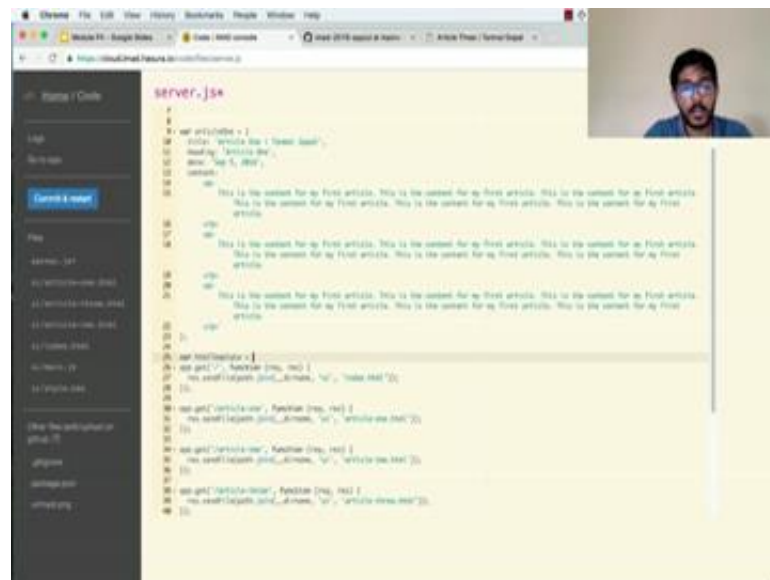


The screenshot shows a code editor with a dark sidebar on the left and a main workspace. The workspace displays the content of a file named 'ui/index.html'. The code is as follows:

```
1 <!doctype html>  
2 <html>  
3 <head>  
4 <meta charset="utf-8" />  
5 <title>My Webpage</title>  
6 </head>  
7 <body>  
8 <h1>Hello World</h1>  
9 <h2>My Webpage</h2>  
10 <div class="container">  
11 <h1>Hello World</h1>  
12 <h2>My Webpage</h2>  
13 <div class="my-button">  
14 <button class="button">Click Me</button>  
15 </div>  
16 </div>  
17 </body>  
18 </html>
```

So, I am going to take this line which is linking the file.

(Refer Slide Time: 27:23)



```
server.js
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

And let us put this into a server js file. Now you can see that as I am getting a lot of java script error because my line is spread across multiple lines; and so in java script you want to have a string that expands multiple line, use back code instead of single code. So, I use a back code and this allows me to write multiline string.

So now we have an object that has content right and this is the content for I am going to call it article one. So, we have we have a content. What we want to do is inject this data into a common HTML template. Let us make another object called HTML template and this is again a multiline string.

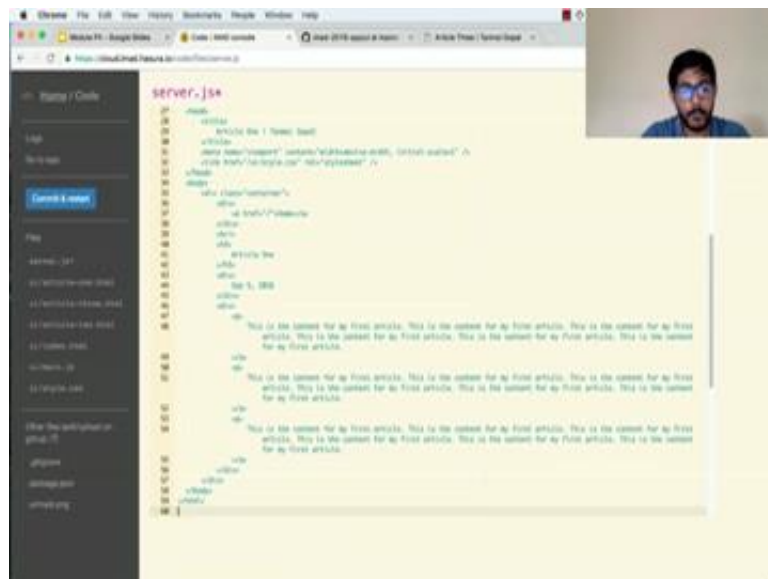
(Refer Slide Time: 28:14)



The screenshot shows a code editor with a dark sidebar on the left and a main workspace. The workspace contains HTML code for a file named 'us/article-one.html'. The code includes a head section with a title 'Article One | News - South', a meta charset, and a link to a stylesheet. The body contains a main heading 'Article One', a date 'Sat 5, 2020', and three paragraphs of placeholder text: 'This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article.', 'This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article.', and 'This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article.' A small video inset in the top right corner shows a man with a beard and glasses.

And let us copy this HTML.

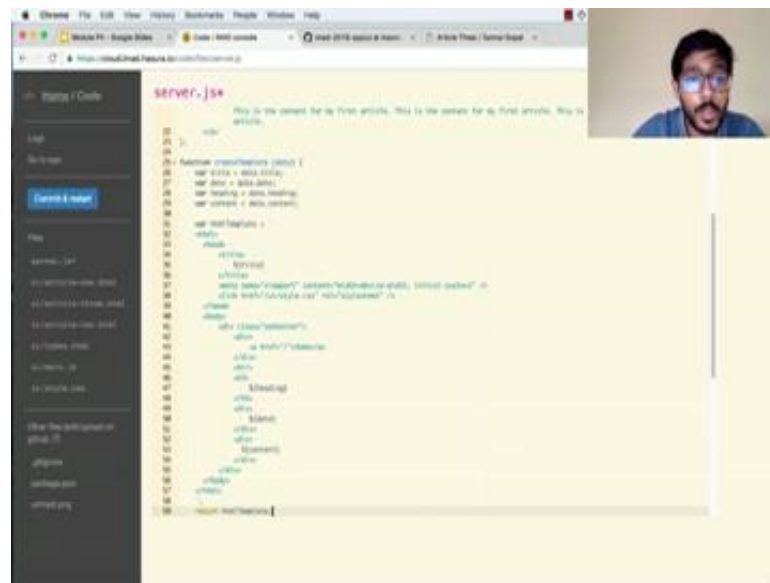
(Refer Slide Time: 28:19)



The screenshot shows a code editor with a dark sidebar on the left and a main workspace. The workspace contains JavaScript code for a file named 'server.js'. The code includes a 'const' declaration for 'app' and a 'listen' method call. The code is identical to the HTML code shown in the previous screenshot. A small video inset in the top right corner shows the same man as in the previous screenshot.

We have to copy that. Let us remove the portions that are common, so I am going to delete this. And I am going to replace this with content.

(Refer Slide Time: 28:46)

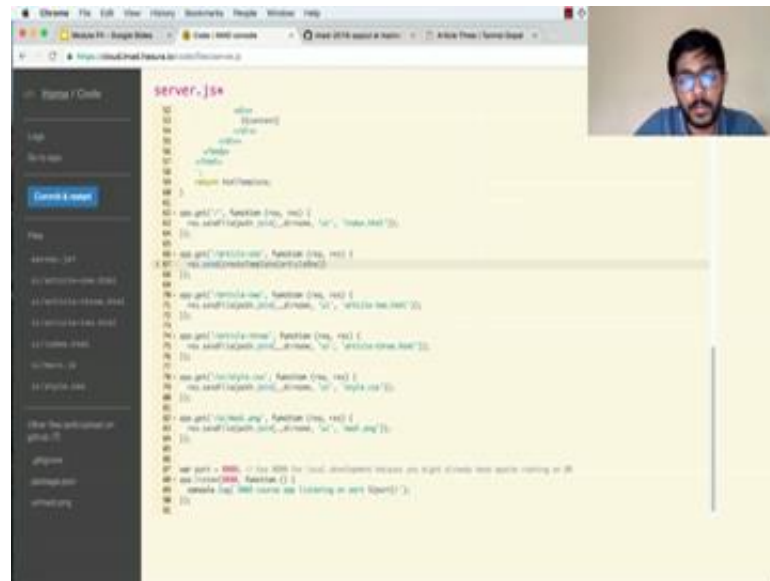


```
server.js
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
50 |
51 |
52 |
53 |
54 |
55 |
56 |
57 |
58 |
59 |
60 |
61 |
62 |
63 |
64 |
65 |
66 |
67 |
68 |
69 |
70 |
71 |
72 |
73 |
74 |
75 |
76 |
77 |
78 |
79 |
80 |
81 |
82 |
83 |
84 |
85 |
86 |
87 |
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
```

I am going to replace this part with the date going to replace this part with heading; I am going to replace this part with title right. So, these are the portions that will come through my java script object, and they injected into the HTML. So, now, this is a special feature of java script that allows us to create a string that has variables in the middle; however, these variables do not actually exist because there is no variable call title heading date and content yet. So, what we actually have to do is create a function called create template and this function will take a data object right.

And let us look at here. So, this is a function that has a data object. And let us create the variables to makes sure that these variables exists. So, we have title which is equal to data dot title, we have date which is again coming from data, we have heading which comes from data dot heading. And we also have content which come from data dot content. So, assuming that the create template function gets an object that contains title, date, heading and content, we will be able to create this string that contains the entire HTML and then lets to return HTML template. So, this will now written by the function.

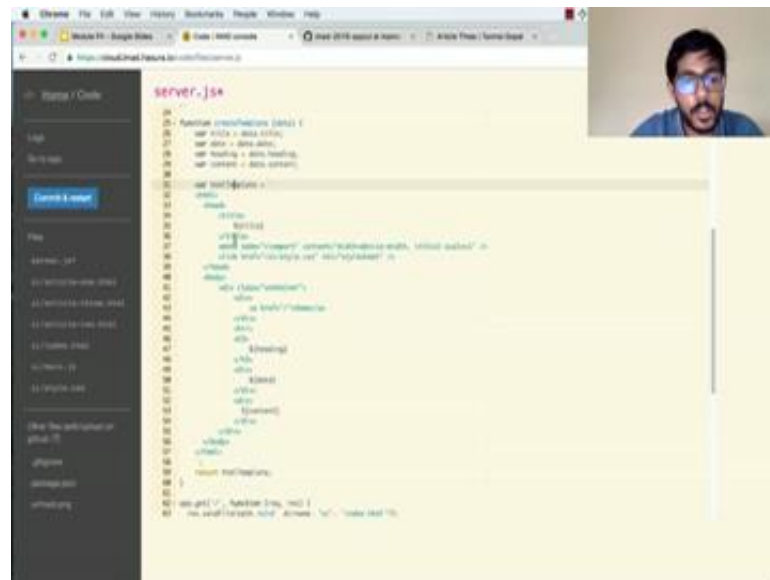
(Refer Slide Time: 30:10)



```
server.js
12 // ...
13 // ...
14 // ...
15 // ...
16 // ...
17 // ...
18 // ...
19 // ...
20 // ...
21 // ...
22 // ...
23 // ...
24 // ...
25 // ...
26 // ...
27 // ...
28 // ...
29 // ...
30 // ...
31 // ...
32 // ...
33 // ...
34 // ...
35 // ...
36 // ...
37 // ...
38 // ...
39 // ...
40 // ...
41 // ...
42 // ...
43 // ...
44 // ...
45 // ...
46 // ...
47 // ...
48 // ...
49 // ...
50 // ...
51 // ...
52 // ...
53 // ...
54 // ...
55 // ...
56 // ...
57 // ...
58 // ...
59 // ...
60 // ...
61 // ...
62 // ...
63 // ...
64 // ...
65 // ...
66 // ...
67 // ...
68 // ...
69 // ...
70 // ...
71 // ...
72 // ...
73 // ...
74 // ...
75 // ...
76 // ...
77 // ...
78 // ...
79 // ...
80 // ...
81 // ...
82 // ...
83 // ...
84 // ...
85 // ...
86 // ...
87 // ...
88 // ...
89 // ...
90 // ...
91 // ...
92 // ...
93 // ...
94 // ...
95 // ...
96 // ...
97 // ...
98 // ...
99 // ...
100 // ...
```

So, I shall go to article one and instead of doing a send file, I am going to change this back to just a send create template, and I am going to give it the content object that I had which was article one right. So, as you can see the article one object is here.

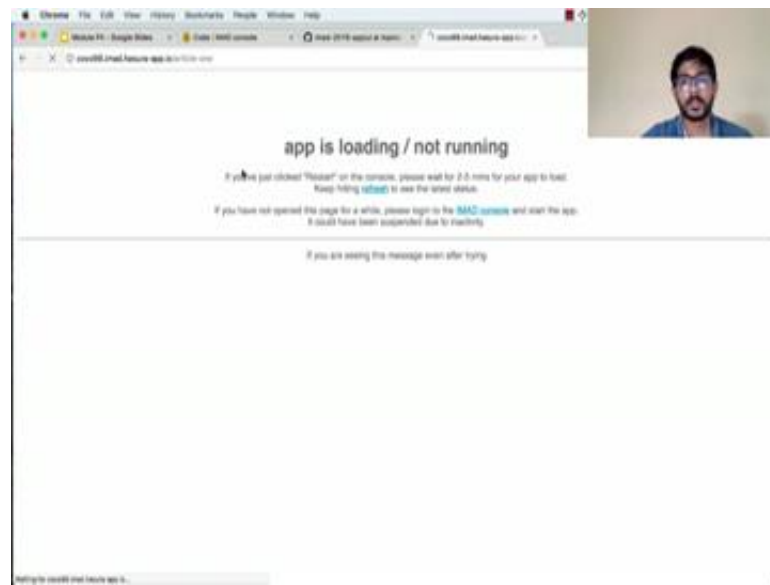
(Refer Slide Time: 30:28)



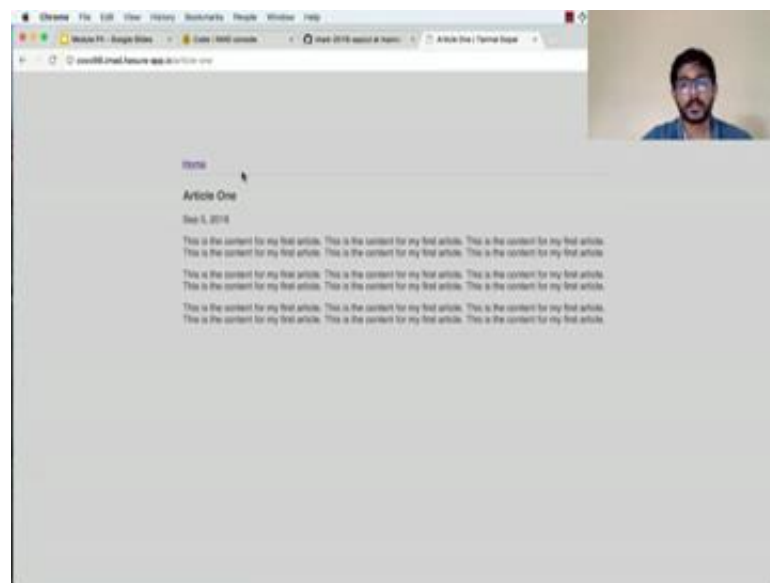
```
server.js
20 // ...
21 // ...
22 // ...
23 // ...
24 // ...
25 // ...
26 // ...
27 // ...
28 // ...
29 // ...
30 // ...
31 // ...
32 // ...
33 // ...
34 // ...
35 // ...
36 // ...
37 // ...
38 // ...
39 // ...
40 // ...
41 // ...
42 // ...
43 // ...
44 // ...
45 // ...
46 // ...
47 // ...
48 // ...
49 // ...
50 // ...
51 // ...
52 // ...
53 // ...
54 // ...
55 // ...
56 // ...
57 // ...
58 // ...
59 // ...
60 // ...
61 // ...
62 // ...
63 // ...
64 // ...
65 // ...
66 // ...
67 // ...
68 // ...
69 // ...
70 // ...
71 // ...
72 // ...
73 // ...
74 // ...
75 // ...
76 // ...
77 // ...
78 // ...
79 // ...
80 // ...
81 // ...
82 // ...
83 // ...
84 // ...
85 // ...
86 // ...
87 // ...
88 // ...
89 // ...
90 // ...
91 // ...
92 // ...
93 // ...
94 // ...
95 // ...
96 // ...
97 // ...
98 // ...
99 // ...
100 // ...
```

So, I have the article one object here; I use the article one object and give it to the create template function, the create template function takes that object extracts the right variables from it, create a string, returns the string and then that string is sent back in a send request. Let us save all of this work.

(Refer Slide Time: 30:50)

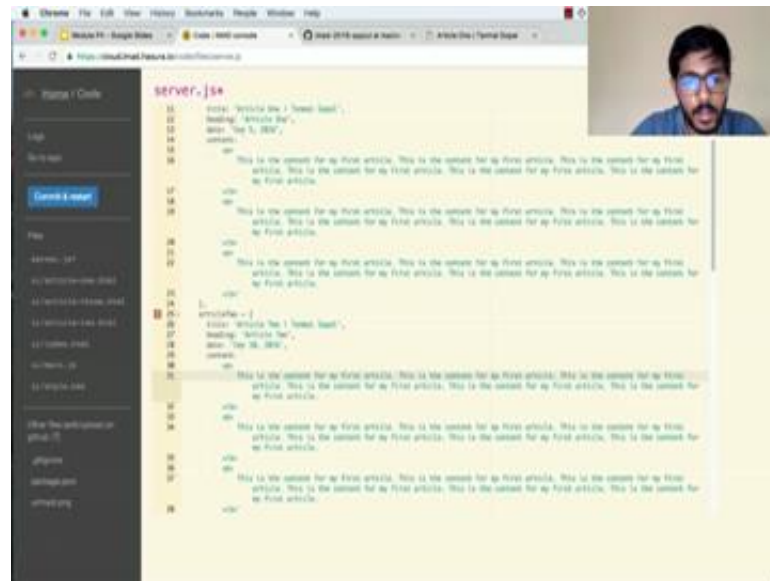


(Refer Slide Time: 30:56)



So, we have article one working. If you notice the background has been applied which was a gray background which was a part of the CSS file that we refer to.

(Refer Slide Time: 32:36)



```
11 const articles = [
12   {
13     title: 'Article One',
14     content: 'This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article.',
15     date: '2020-10-01'
16   },
17   {
18     title: 'Article Two',
19     content: 'This is the content for my second article. This is the content for my second article. This is the content for my second article. This is the content for my second article. This is the content for my second article.',
20     date: '2020-10-02'
21   },
22   {
23     title: 'Article Three',
24     content: 'This is the content for my third article. This is the content for my third article. This is the content for my third article. This is the content for my third article. This is the content for my third article.',
25     date: '2020-10-03'
26   }
27 ];
28
29 const getArticles = () => {
30   return articles;
31 };
32
33 module.exports = {
34   getArticles
35 };
36
37 const { getArticles } = require('server.js');
38
39 console.log(getArticles());
```

So, now what we would like to do is write this data, copy this data; and said article two right we can change our content here and here and change the date to may be 10th to reduce the lot of content on this that it says the content for my second article right. So, pay attention to the error block here, where it saying that my syntax is wrong. And it should not be an equal to, it should actually be a colon because we inside an object right.

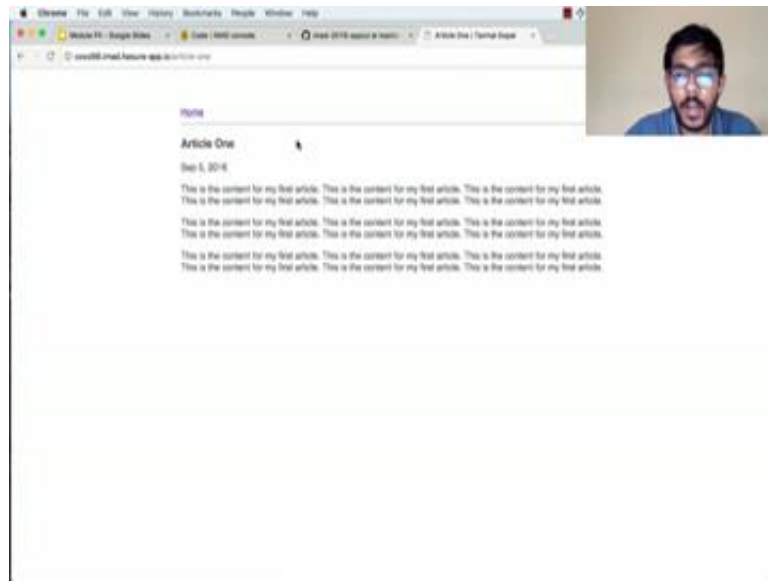
(Refer Slide Time: 33:10)



```
11 const articles = [
12   {
13     title: 'Article One',
14     content: 'This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article. This is the content for my first article.',
15     date: '2020-10-01'
16   },
17   {
18     title: 'Article Two',
19     content: 'This is the content for my second article. This is the content for my second article. This is the content for my second article. This is the content for my second article. This is the content for my second article.',
20     date: '2020-10-10'
21   },
22   {
23     title: 'Article Three',
24     content: 'This is the content for my third article. This is the content for my third article. This is the content for my third article. This is the content for my third article. This is the content for my third article.',
25     date: '2020-10-03'
26   }
27 ];
28
29 const getArticles = () => {
30   return articles;
31 };
32
33 module.exports = {
34   getArticles
35 };
36
37 const { getArticles } = require('server.js');
38
39 console.log(getArticles());
```

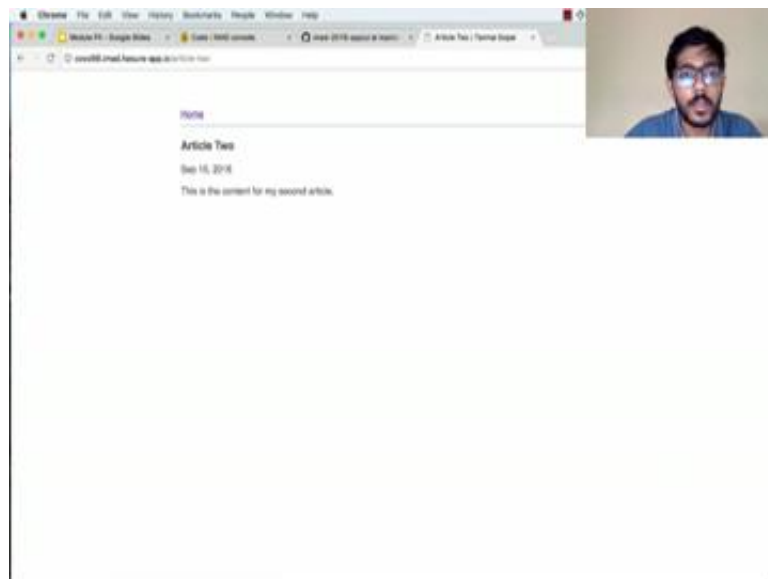
So, I am going to take this for article 2; and make this article 3; article 3 and 15 is the content on third article.

(Refer Slide Time: 36:05)



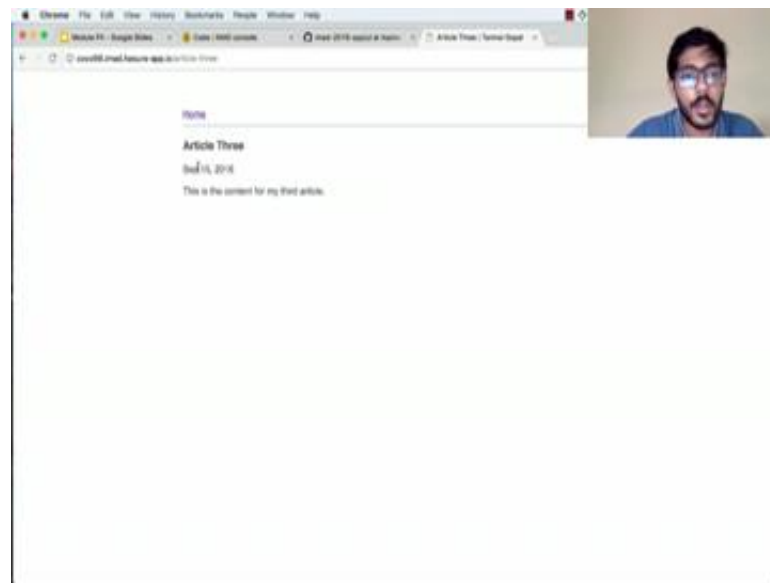
So, I have article-one.

(Refer Slide Time: 36:08)



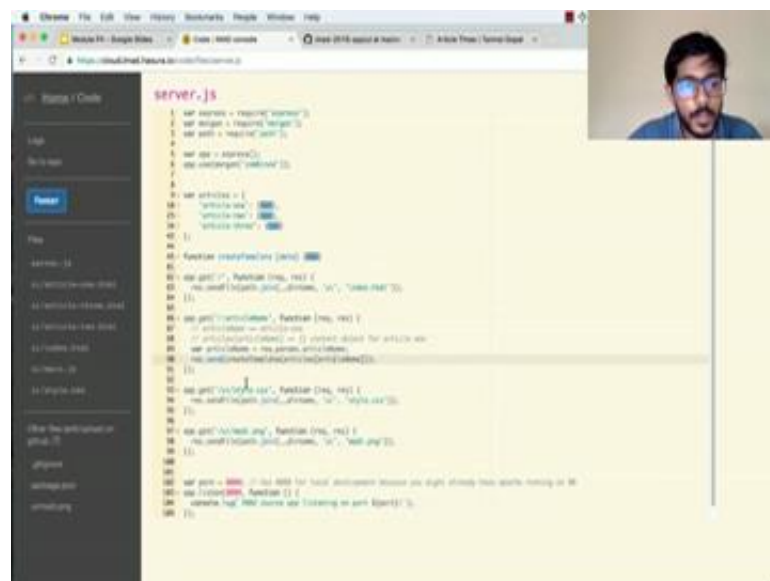
That loads that is changes to article-two. Article-two also working and you can see the new date.

(Refer Slide Time: 36:13)



And let us go to article-three and we can see that the article-three is working as well.

(Refer Slide Time: 36:16)



And we reduce the lot of code from our server dot js file and we actually do not need these files anymore.

(Refer Slide Time: 36:31)



(Refer Slide Time: 36:34)



Let us say commit lets go to article-two, delete article-three, we would not need this file anymore. Refresh our console.

(Refer Slide Time: 37:08)

Press (Esc) to exit full screen

Lots of stuff to learn!!

References to help you understand what we did more clearly:

- 1) HTML `<style>` tag for writing CSS inside the HTML
 - <https://developer.mozilla.org/en/docs/Web/HTML/Element/style>
- 2) CSS syntax for class names (not just HTML elements)
 - https://developer.mozilla.org/en/docs/Web/Guide/CSS/Getting_started>Selectors
- 3) `<link>` tag to link the CSS file to an HTML document
 - https://developer.mozilla.org/en/docs/Web/HTML/Element/link#including_a_stylesheet
- 4) CSS syntax for adding styling to a page
 - https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started
- 5) Javascript objects and multi-line strings using backquotes (`)
- 6) Javascript functions and template strings
 - https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Template_literals
- 7) URL parametrization in express routes
 - <http://expressjs.com/en/guide/using.html#route-parameters>

Introduction to Modern Application Development | Dr. Gaurav Rana (IIT Madras), Tanmay Gupta (Harvard)

So, this was an extremely heavy module because not only that we take a lot of our concepts and apply them, but we also wrote a lot of specific code. And there are many times in this video, when you would have not seen these different programming elements and so I have listed out a bunch of references that you guys should refer to and use to understand the kind of step that we did.

Just too quickly read those out we use the style tag for we use the style tag, some CSS, and you can read up little more about how CSS selectors work. The CSS selectors what allows us to select the right HTML element is to modify. We also use the link tag to link our CSS; we can actually use a link tag to link many other things apart from a CSS. We should also read up a little bit about CSS and this guy from Mozilla is a nice place to read up on CSS.

we also used multiline strings and template strings which is the new java script language feature and gives us back quotes. The back quote characters typically to the left of a one on your keyboard, and you might not have used it very often. We also used URL parameterization, which is a way to extract a part or a route or URL in express and convert that into a variable that can be used in code. So, read upon these things to help you understand how we wrote the code that we wrote today.

Also, while you are writing code you might have to iterate quite frequently, because very often you will be making small, small mistakes that you will only catch after you deploy

a code. So, I would recommend that you actually try to set up the local development environment, because that will help you iterate much faster. Please refer to the previous module to understand how to set up a local development environment.

There are lots of guides in the internet to help you set up the environment on Linux, or Windows or Mac and use those guides in fact just as a quick note. A lot of time when you do application development, most of the latest information is something that you will have to find yourself online.

And for example, the reference is that I gave for installing something on Mac is something that might change 6 month later or 1 year later and because these things are moving so quickly. We as developers have to get use to searching for the content on the internet, used to collaborating with a community and reading and learning by ourselves on the internet. So, please do not give up or lose hope and things are not working for you, if you are not able to set get up or if you are getting functions errors, do a lot of research, be persistence, search for things in Google, ask questions in the forum and you eventually get your environment working for a you.

(Refer Slide Time: 39:47)



The slide features a black header bar at the top. Below it, the text "Coming up..." is displayed in a large, bold font. To the right of this text is a small, square video inset showing a man with a beard and glasses. Below the main text, there is a bulleted list with two items: "Introduction to client side Javascript" and "Write our own API". At the bottom of the slide, there is a dark grey footer bar containing the text "Introduction to Modern Application Development" on the left and "Dr Gaurav Raina (IIT Madras), Tarun Gopal (Hassara)" on the right.

In the next module, we are going to be learning what client side java script, which is java script that executes on the browser not java script that is executing on the server. And we will also learn how to write our own API, and we will actually implement our own API end point.