INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

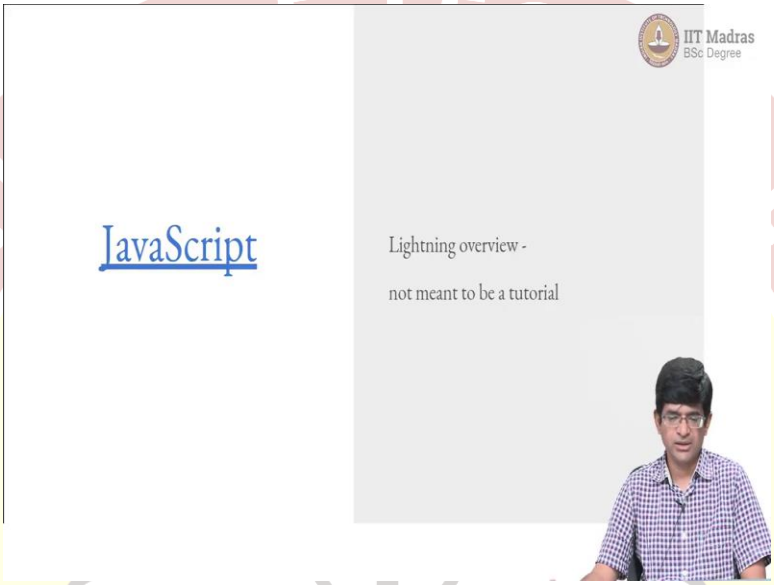
IIT Madras

ONLINE DEGREE

Modern Application Development
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
JavaScript

Hello, everyone, and welcome to this course on Modern Application Development.

(Refer Slide Time: 00:18)



So, what we are going to do now is to have a sort of lightning overview of the topic of JavaScript. So I want to be very clear about one thing. This is not meant to be a tutorial on JavaScript. It is not sufficient to really get you to the point where you can write code in JavaScript. I will not even be showing you how to run a JavaScript piece of code in a browser, for example.

But JavaScript as a language is sufficiently similar to other programming languages that you can pick up, the syntax itself through other means. The main focus over here is going to be on sort of what are its capabilities, what can you do with it and why is it needed in the first place.

(Refer Slide Time: 01:02)

So, let us take a moment to understand what is JavaScript, to start with. So if you go look at the Wikipedia page for JavaScript, for example, you will see that it is a high level language. What does high level language mean? For those of you who have done some amount of programming you would know that you can have languages at different levels.

In particular there is the so called assembly language, which is used, which is pretty much the direct machine instructions used by a processor. Above that you have languages like C, which are usually called low to intermediate level languages. So C is not really a low level language, assembly is a low level language. C is still called a high level language but it definitely ranks a bit lower on the scale than some other languages like Python for example.

Why is that? Because there are many things that are very strict about C. I mean it allows you to only, you have to declare variables in a certain way, you have to, you are directly sort of manipulating pointers in the memory of the system and so on. The good part about it is it gives you fine grain control over what is happening inside the system.

The bad part is very often you should not have that kind of fine grain control because it is not good, it does not play well with an operating system it makes it difficult for someone else to write with, work with your program and so on.

A higher level language would be something like Python or MATLAB where we are talking about much higher levels of abstraction. I can actually think about arrays or lists as being a first class data type in Python. I do not even think about int versus float and so on. They are all there inside Python but I just talk about numbers.

I can declare a variable or rather I can use a variable without even needing to declare it, and the Python interpreter will pretty much figure out what type it is and do whatever is needed in order to make things work. So that something called dynamic typing.

So, Python, for example, is a high level programming language it also has object orientation. You have this concept of classes and so on. JavaScript in that sense is very similar. It has dynamic typing it has object orientation.

This object orientation in JavaScript is based on something called prototypes rather than the regular class based inheritance that we talk about. It is not necessary to know too much detail about that to start with. If you are interested, of course, you can read more about it but it does not make too much difference to how you use the language unless you start getting into details of that.

So, the bottom line is it is a high level programming language. JavaScript, similar to Python, has concepts of lists or arrays, it has dictionaries so that you can have what are called associative arrays or dictionaries or maps, which make it very easy to create really complicated data structures.

So, once you have that it means that you can very easily start constructing things like trees or graphs or various other things just by having different kinds of objects referencing each other.

Now in addition to that the Wikipedia page also says that JavaScript is a multi-paradigm language. So what does multi-paradigm mean? A paradigm is a way of thinking about things. And what that means is a programming language like C usually falls in this class of languages that are called imperative.

And imperative means that you pretty much write out functions to do each and every step that you want, and you call those functions one by one, you basically specify each of

them in the order in which it is to be done. So a C program you pretty much read it from top to bottom and that is pretty much the order in which it gets executed.

Now that is essentially one way by which you can use JavaScript. You can directly define functions, you can define procedures, you can call them one after the other, and yes, JavaScript will work fine that way.

On the other hand, there is also this concept of what is called functional programs, and in functional programs they are based on sort of the mathematical theory of functions. There are a whole bunch of languages that are designed around this. One of the most famous original ones was called Lisp. And nowadays you might come across terms like Haskell or Ocaml, which are also strongly functional languages.

Now it is not just that those languages are a fact, they fundamentally change the way you think about the problem. So rather than just sort of writing loops to sort of perform computations, you try and write the mathematical equations that define a certain computation and the compiler or the interpreter then figures out the best way of executing it, which means that from the programmer's point of view, potentially, there are like really powerful ways of looking at the problem and leaving a lot of the grunt work to the compiler.

And more importantly having some knowledge of functional programming can actually change the way you even think about a program or how you to solve a problem. So that functional is also one of the ways in which you can look at JavaScript programs.

Why is that? Because you can do things like you can essentially treat a function as a first class object in JavaScript. You can basically assign it into a variable, you can pass functions around, you can create a function of a function, higher order functions and so on, composition of functions, so all of that allows you to do some slightly more advanced type of coding than you would otherwise do.

Once again, do not get into this just for the sake of doing it. You need to sort of try and understand what is meant by functional programming first before you get into trying something of this sort with JavaScript or any other language.

And one other important paradigm as far as JavaScript is concerned is this notion of being event-driven. Now event-driven basically means that you respond to events. So you can have different functions, different parts of your code which are all present and available, but rather than calling all those functions one after another what we say is if something happens then, one function will get called, or if something else happens another function will get called.

Now this is pretty much perfect for GUIs in general or for interacting with let us say a webpage because what happens on a webpage, I display the page and after that I am waiting for user input. If the user clicks on a button, call a specific function, if they click on a link, call some other function, if they click on another button, call a third function, if they select some part of the text, maybe call another function.

All of those are events. Clicking, selecting, all things of that sort are events that the user is providing to the program. And rather than sort of going on saying okay, what did the user do, what did the user do, what did the user do, you let the functions be bound to certain events. And anytime the event happens automatically the system takes care of calling the appropriate function.

A very powerful way of thinking about reactive systems, that is to say interactive systems in general, and fits the model of the web perfectly. JavaScript also happens to be relatively easy to learn. Especially if you have familiarity with something like Python, you should not find JavaScript too difficult.

Now, one thing to keep in mind is except for the name, it has very little connection with Java. It came about because Java was brought in as a language that could be used to create applets that would run inside browsers. And JavaScript basically said forget applets we will just run directly inside the browser. And that is the only real connection between the two.

There are some syntax that are vaguely similar between the two in a few places, but you will probably find that there is more similarity between JavaScript and Python than JavaScript and Java, so to say. So there is no direct relationship. So it is not as though knowing Java will help you with JavaScript. You need to think of it differently, and you need to understand why it is relevant for use in the context of the web.

(Refer Slide Time: 09:36)

So, why is JavaScript useful? The main reason is that most web browsers today have a dedicated engine just for running JavaScript programs. It is designed from the ground up, the language JavaScript was designed from the ground up for the web. Now why do most web browsers have this engine? Because historical reasons.

People found that JavaScript was useful therefore a web browser without JavaScript would not see as much popularity, so everybody started adding JavaScript to their browsers, over time it became a necessity, you had to have it or else nobody would use your browser.

Now the great thing about JavaScript is, actually speaking, it is an interesting language, it does not have any notion of input output, unlike, let us say, Python, where you can directly read from a file or write to a file. In JavaScript you do not have the basic capabilities for doing that.

The reason being on the web you do not really have a way of getting input or giving output directly to a user or through files. On the other hand, there is, I mean so there is no native IO, no file access etc, but even that can be provided through the use of APIs. So everything about JavaScript ultimately comes down to APIs.

And the browsers tend to support APIs for a very large number of different kinds of functionality. So this is not part of the core JavaScript language, but in general, it is

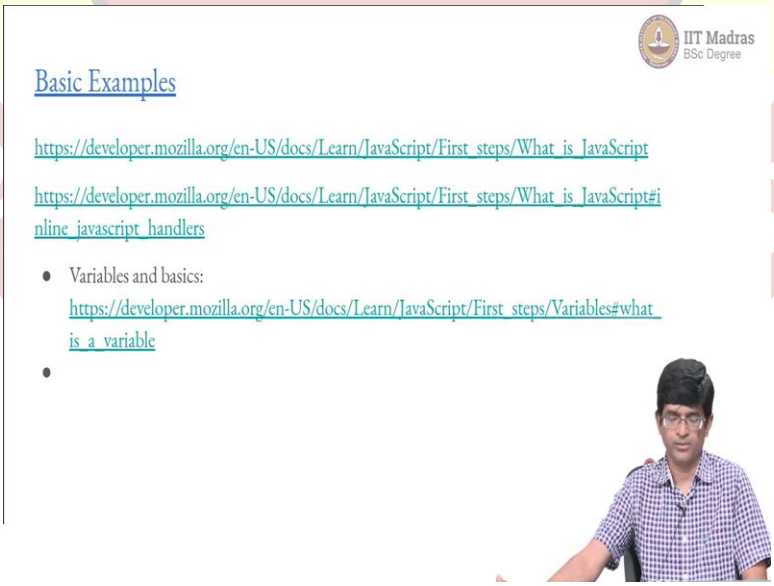
something that you can expect that a JavaScript runtime, that is to say, JavaScript in a browser, is most likely going to support.

So, manipulating text, strings, dates, regular expressions which are used for various kinds of pattern matching, which you may have heard of, you may have used in different contexts but sort of beyond the scope of what we have here, this is supported again, using APIs. It has many standard data structures, dictionaries, lists and so on, which you can directly use, and you, using those you can create much more complex data structures.

A very important part of JavaScript is the API for manipulating the DOM, the document object model, which basically means this is ultimately how you are manipulating the browser, you are able to directly modify what shows up on screen. And that is a large part of where the power of JavaScript comes.

At the same time, like I said it does not have native input output capabilities, no file access for example, but even that can be provided through APIs, and there are ways by which the browser can give you controlled access to different parts of the file system. Now the maximum power of JavaScript, of course, comes when it is used for DOM manipulation and that is where you will find the majority of JavaScript code being used.

(Refer Slide Time: 12:27)



So now, what I am going to do is to actually sort of look at a few examples that are already there on the web because ultimately the best way to sort of learn JavaScript is by writing code.

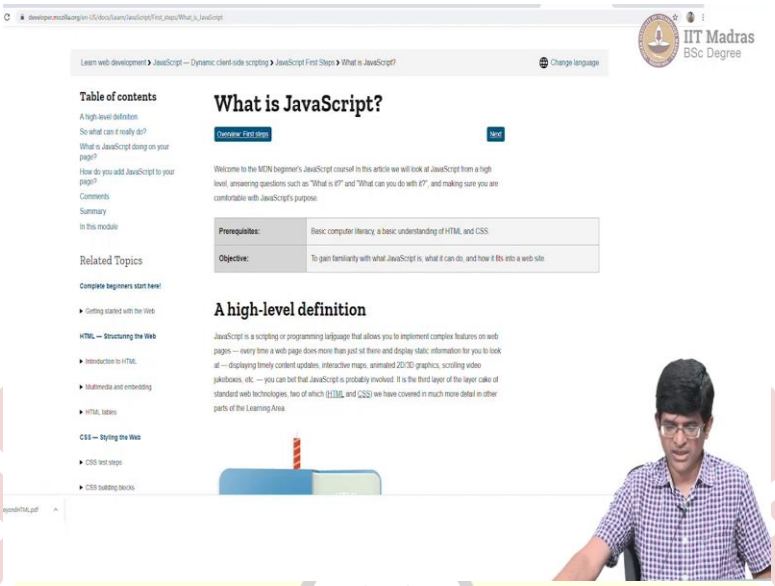
So, there is no point in giving you a complete sort of overview of what is the syntax, what are the ways in which you can write JavaScript and so on. You should probably do that if you are interested, and go through, there are any number of resources online that would help you with this, but the thing to keep in mind over there is that which one do you go to.

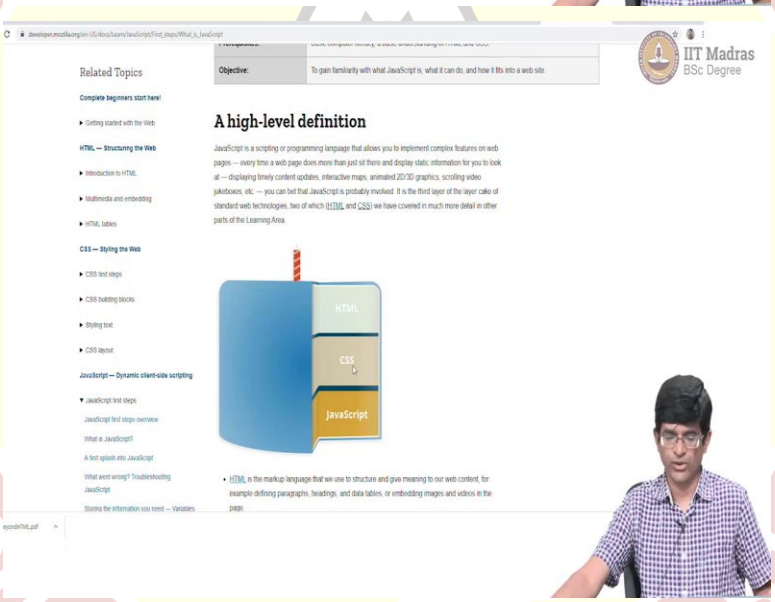
One generally good set of links is provided by the Mozilla developer network, so developer.mozilla.org or there is also MDN, Mozilla developer network. Why Mozilla? Well, Mozilla makes Firefox. It is one of the most powerful browsers, most popular browsers. Maybe not in terms of share numbers at this point but at least most people know of Firefox.

And the reason for that is because the Mozilla, that group has been involved with web related technologies pretty much from the start of the web. The very first web browser that was sort of publicly used was Netscape navigator, and that is where Mozilla has sort of evolved from.

So, they have a lot of useful information on their developer website, and what I am going to do is just look at a few examples. So like I said I cannot go into the details of the syntax of the language or give you too much information about how to use it. So this is more sort of first steps, just a flavor of the language. Hopefully it will get you interested enough to learn more of it on your own.

(Refer Slide Time: 14:22)



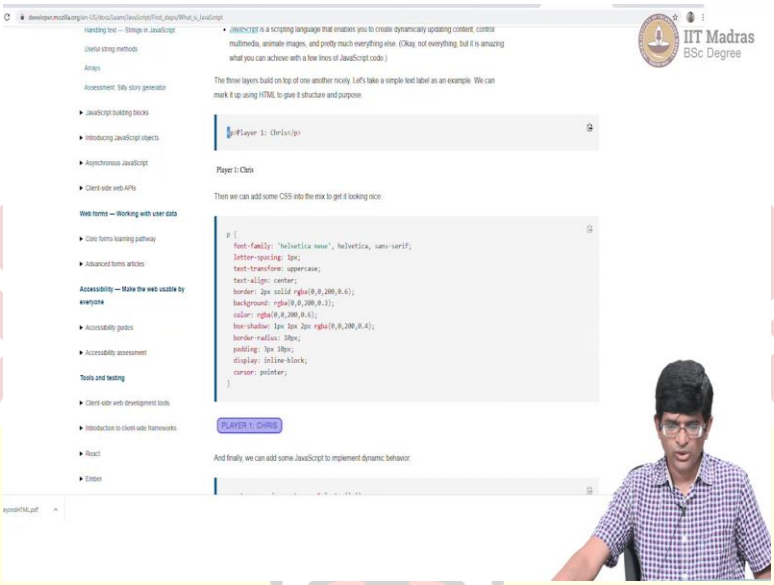


So, the, what is JavaScript page on the Mozilla developer network? It gives you a lot of information, it talks about high level definition of JavaScript, scripting or programming language, etc.

The interesting thing is the way that they look at it, it essentially looks at, at html, CSS and JavaScript as sort of, I mean they talk about it as three layers of a cake, but ultimately these are the three sort of defining technologies of the web as we know it today. It is not that this is something fundamental. It is just that over the past several years it has grown to be, to this point. And these are the main sort of pillars of the web as it stands today.

Html and CSS, you have a reasonable understanding of, at this point. JavaScript, like I said, a full, sort of in-depth discussion of JavaScript is beyond the scope of this course but what we are going to do is at least try and get a flavor for it.

(Refer Slide Time: 15:22)

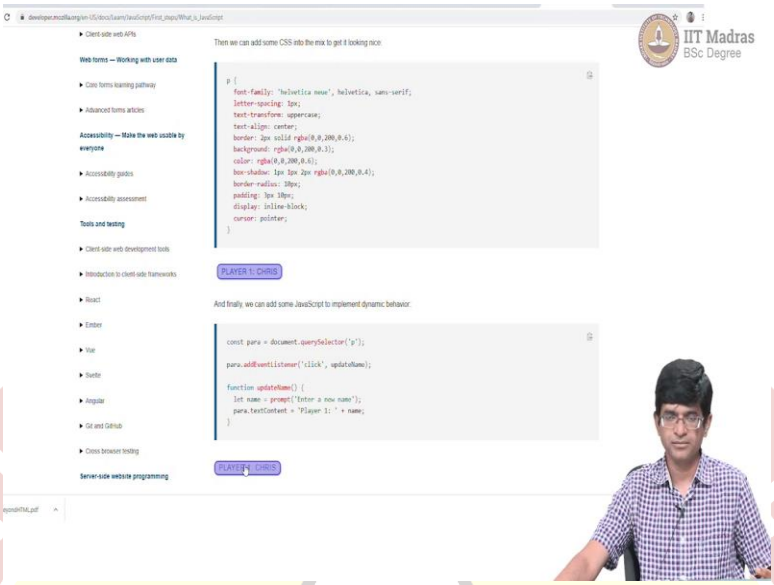


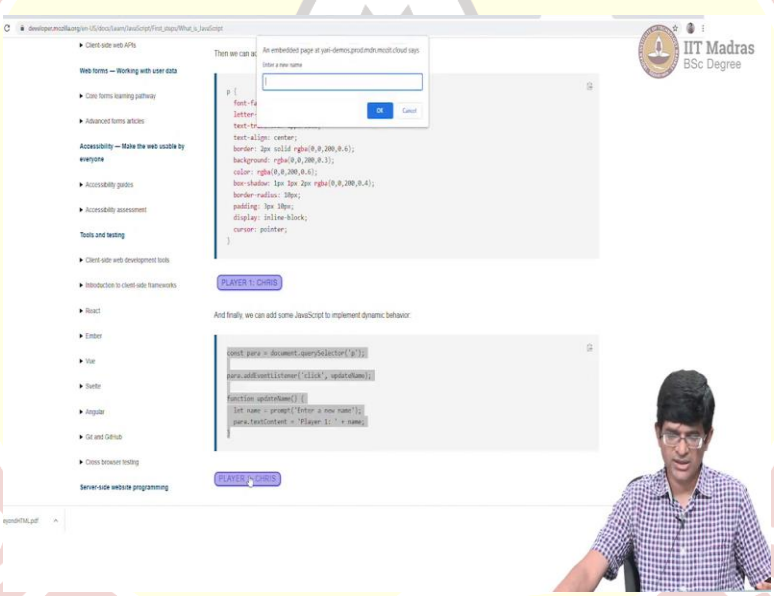
So, let us take a look at this example. So what we have over here is, you have one small piece of html code, with just one paragraph, , and it basically says some text inside that, Player 1 : Chris. And then you add some CSS styling to it, which basically says what font family to use, some Helvetica something, letter spacing.

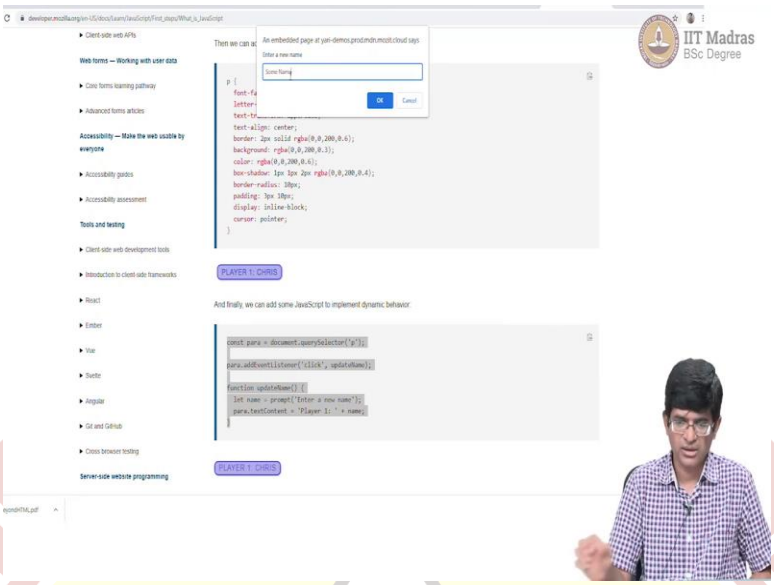
Transform the letters to upper case, which means that even though the name was given as Chris, it will be typeset displayed as capital letters. Align in the center, create a two pixel solid border, make the background some kind of I guess a purplish color.

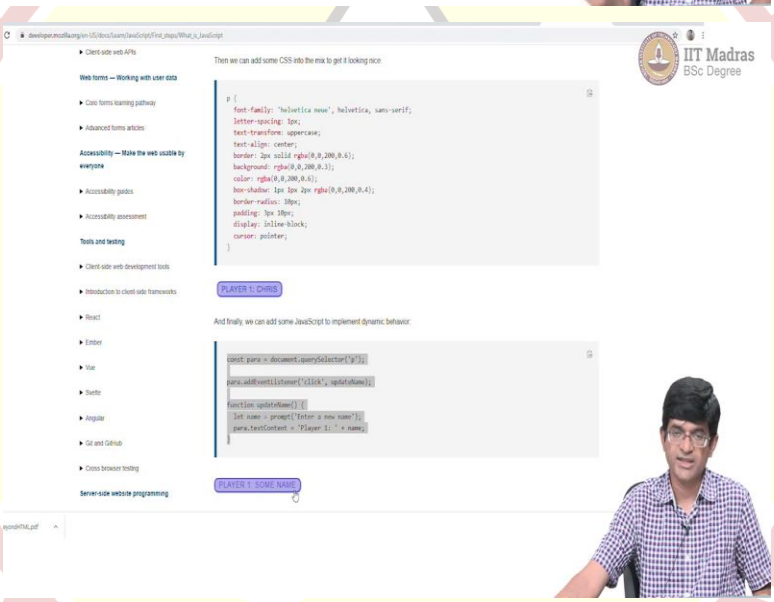
So, all of this information, bottom line is this is what it looks like, this Player 1 : Chris, that you can see here, this sort of violet color display that is there towards the bottom of the screen. So what do we have? We had html and we had css. Combining those two gave us this display.

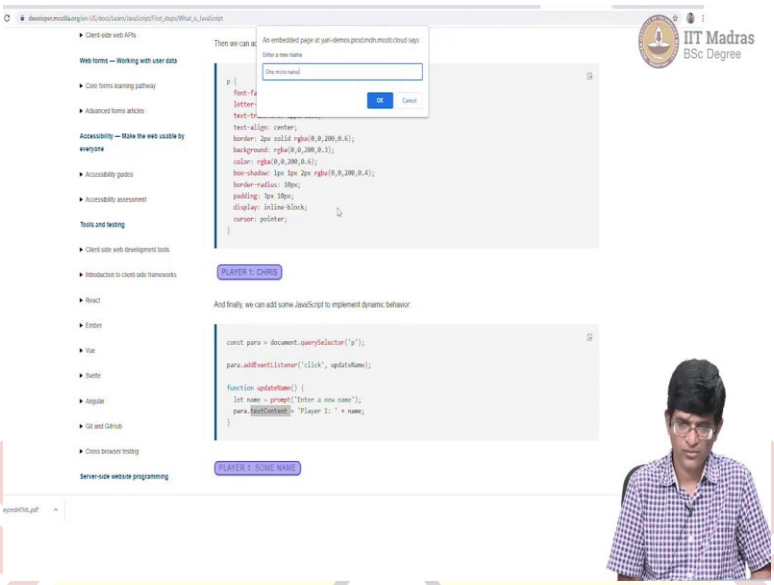
(Refer Slide Time: 16:28)

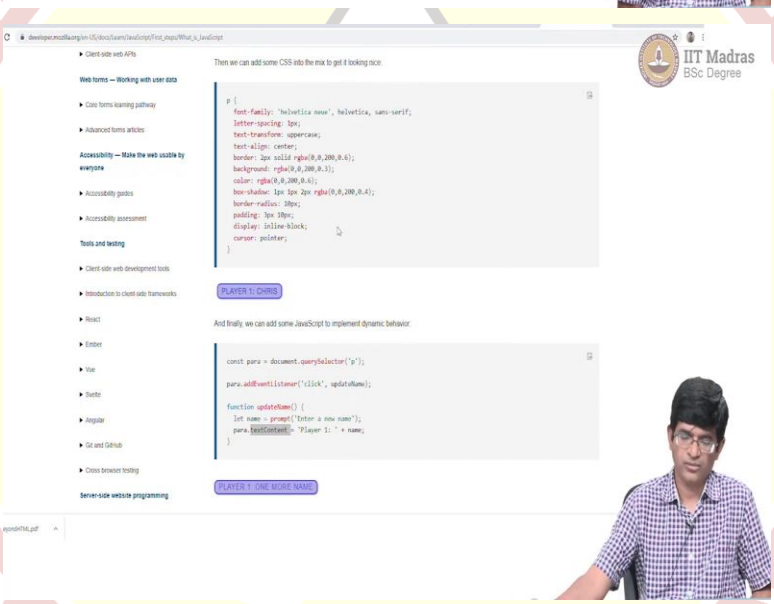












And now, we add a little JavaScript in order to implement dynamic behavior. So what does the JavaScript look like? Let us first take a look at the language itself. So we have one line out here that says const. So const, it is sort of like a constant declaration. The const int that you could declare in a language like C.

Essentially what you are trying to say is that para is something that you want to use like a variable but on the other hand you do not want it to be changed later. That is, in other words, once you have declared para, I do not want it to ever change anywhere else in my code, which is why you declare it as a const.

What happened if you did not declare it as a const? Nothing much. It will still work. The point is somewhere else in your code you might accidentally write para equal to something else and then you might change things and not be able to debug what is happening.

By declaring it as const at least the engine knows that you do not want to change it in future, and probably it will signal some kind of an error, maybe not on the web page but at least somewhere internally.

Now, so that means that para essentially is a variable. It contains some pointer or something. Now, how are we assigning a value to it? We call a function. What function is it? There is an object called document which is part of the basic JavaScript. And document.querySelector, this is exactly like you would do in Python, so there is an object and query selector is a method on that object.

So, document.select, querySelector('p'), in other words, what it is going to do is it is going to select the p type element, a paragraph element present inside the document. And now look at that. So what it has done is? It has basically now got a variable called para which has got assigned whatever came out of this function call. So the document.querySelector is a function call. A method call on an object which is basically ultimately a function call.

Now, what type does it return? Does it return a number? Does it return a string? What it actually returns is some kind of, you can think of it as a pointer, or a handle. That is a different sort of term that we use. Ultimately the point is this itself is a object that comes back, which means that it has its own methods associated with it, which means that I can call something like para. something which is a method associated with the para object now.

So now para, in other words, hopefully after the first line has run, has selected some paragraph object that is present, and para.add event listener is now going back to, you remember we talked about paradigms, this is the event-driven paradigm. What it is saying is, if you perform a click event, click is one of the, sort of predefined events inside JavaScript, call this function updateName().

So, add event listener basically says that if the click operation is performed on the para, which after all came about as the result of query selecting on the p tag, if, in other words, you click on a paragraph, call the function updateName().

What is this function update name? Here, this is how you define a function. In Python you would have used the word def, over here you use function. In Python there would have been a colon followed by indentation and so on. Instead of that, this is a bit more like C or Java. It uses curly brackets to start and end the function.

So, function updateName(), simple enough. Let, let is a different way of again defining a new variable. It is essentially what, let does is it means that it is now scoped meaning that the, this variable name exists only within this function. And once you go out of the function there is no longer a variable called name. You can get into scoping later when you actually start programming. For the time being, it is just a variable.

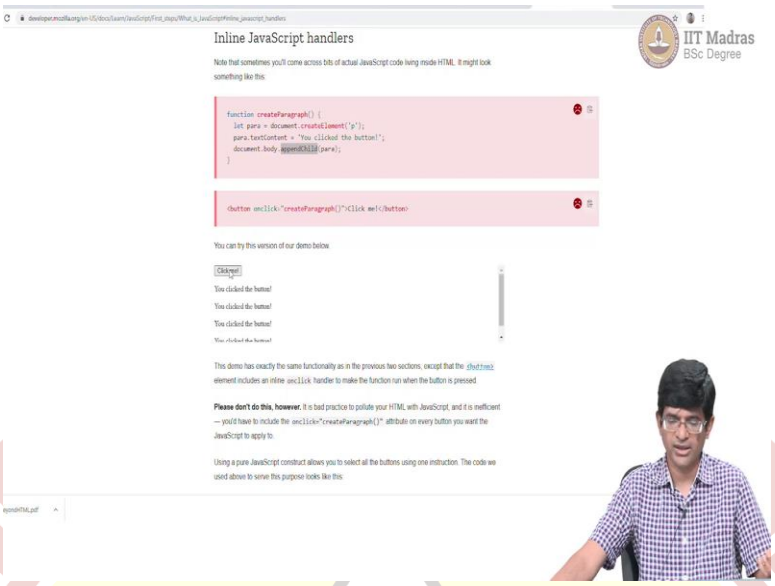
Name = prompt, which means that it should pop up something on the screen asking you to enter a new name. And once you have entered that name the para.textContent, so this is not exactly a function call, this is more like an attribute. So the para.textContent will change to Player 1: whatever you enter over here. And they have got that right here.

So, what has been displayed over here is a paragraph, exactly similar to this Player 1 Chris, that was there earlier, but now with this JavaScript code added to it. What happens when I click on this? It pops up this query which says an embedded page blah, blah, blah says enter a new name.

And what do I do? I can basically enter some, some name. So I enter some name, click OK, and immediately the text changes to Player 1 : SOME NAME. So what happened? This para had selected the p type html element, it added an event listener to it, the event listener would call the function update name anytime that a click happened on that para. And update name would ask you for a new name and update the text content.

I can do it again. I can go here and type one more name, and it changes accordingly. So this, in other words, is a very simple and clean way of adding functionality to a web page. So you can already see the power of JavaScript in the context of the web coming in.

(Refer Slide Time: 22:40)



Now, let us look at another example of JavaScript being used for some additional functionality. Right now, we used it in order to replace the text content of a paragraph. You can do a little bit more. You could actually have some kind of JavaScript code which is there right inside your html as part of a script which is somewhere there.

You define a function, which basically says create paragraph. What does it do? It creates a new element, a p element, sets the text of this, and to document.body, so document remember, was the object corresponding to the entire document, document .body is part of that DOM, the document object model, which says it is the body content of that particular tag, of the particular document, appendChild(para).

So now normally if I run this in the context of this web page, the document body should be the entire web page. But the way that this has been written on this particular website is that this text that you have over here is sort of an, what they call an Iframe, an inline frame. It is a separate webpage of its own, which is just loaded and displayed over here, and just contains this function and this piece of text over here.

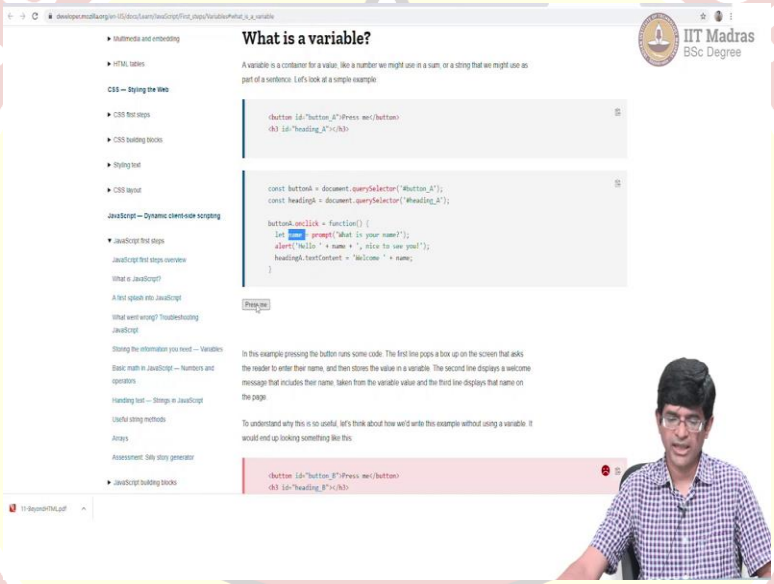
So, what does this piece of html do? It generates a button, which on click will call this function createParagraph(), and the text of the button says click me. So it basically looks like this.

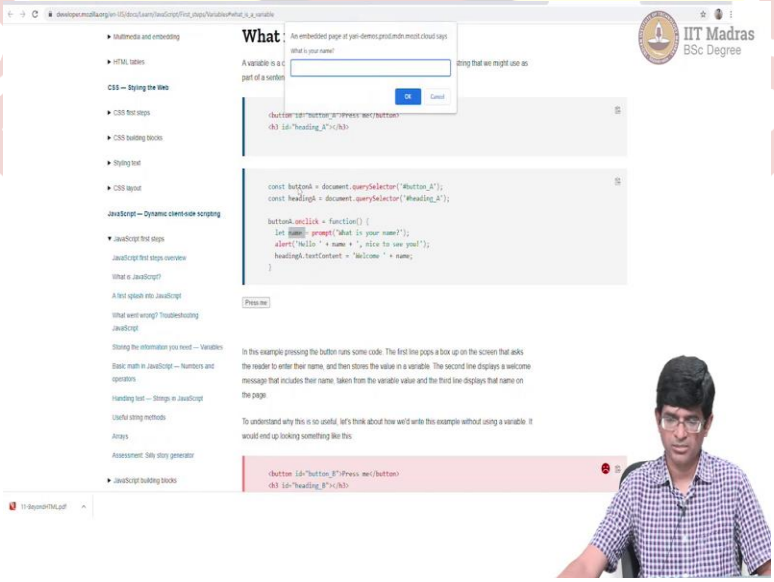
So, the bottom line is, now what we are seeing over here below this, you can try this version of our demo below, immediately below that is something which is actually a

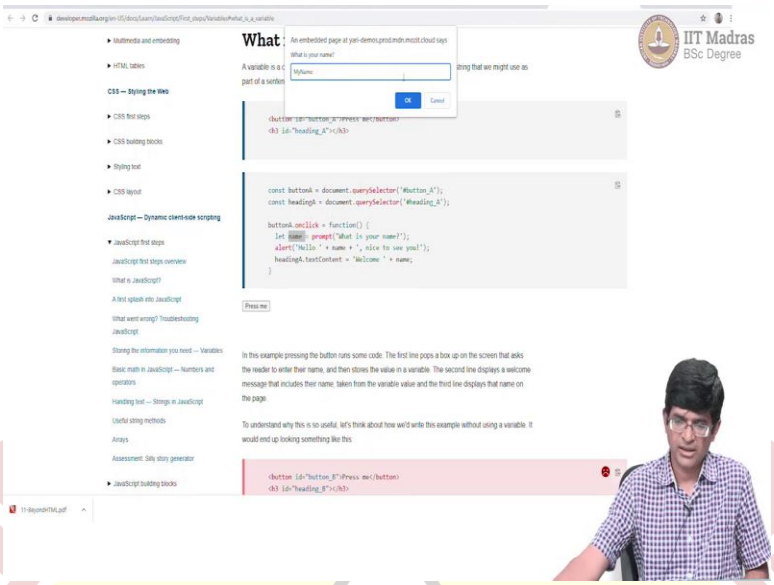
separate web page altogether, because otherwise if I went and clicked on this, it should do this append to the entire document, which means that at the bottom of this webpage, and it might not even show up.

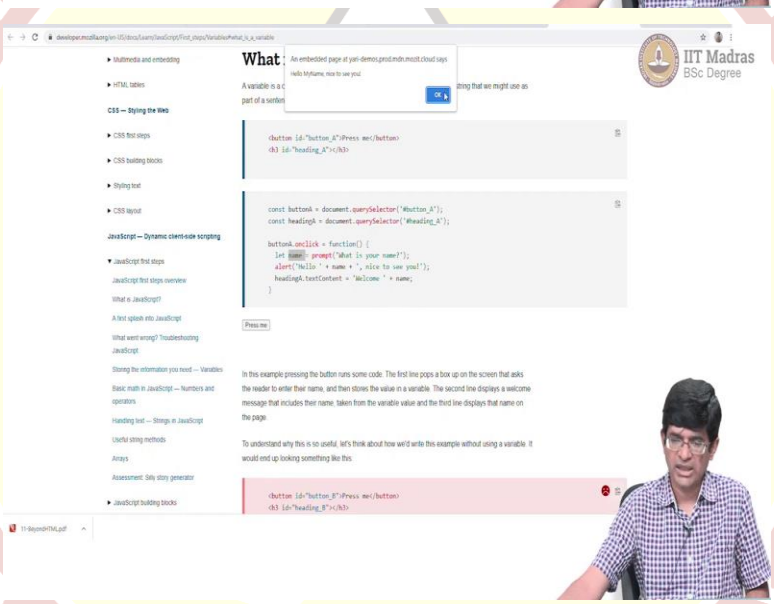
On the other hand, because it is doing it in this Iframe, whenever I go and click here, it says you clicked a button, you clicked a button, you clicked a button. I can keep doing it and after sometime it keeps on adding more and more text out here. I can do this as many times as I like. So, these are sort of instances of how you can get basic JavaScript, and add functionality to a webpage.

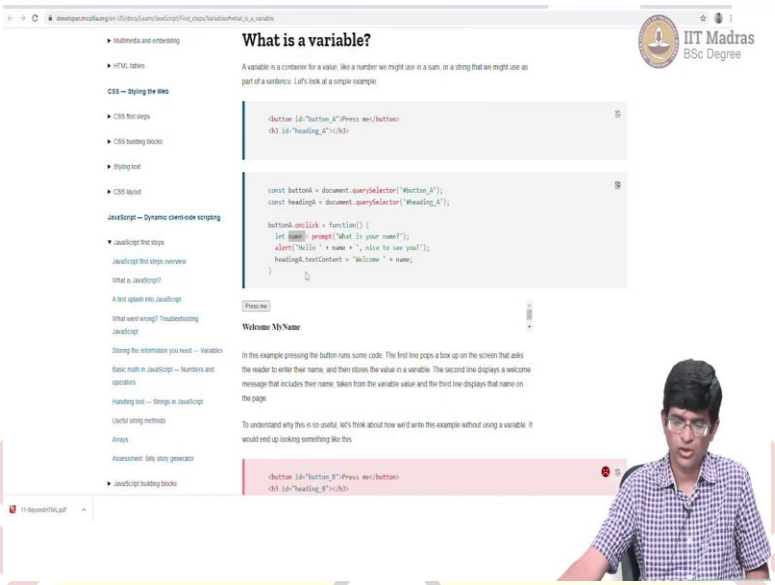
(Refer Slide Time: 25:05)

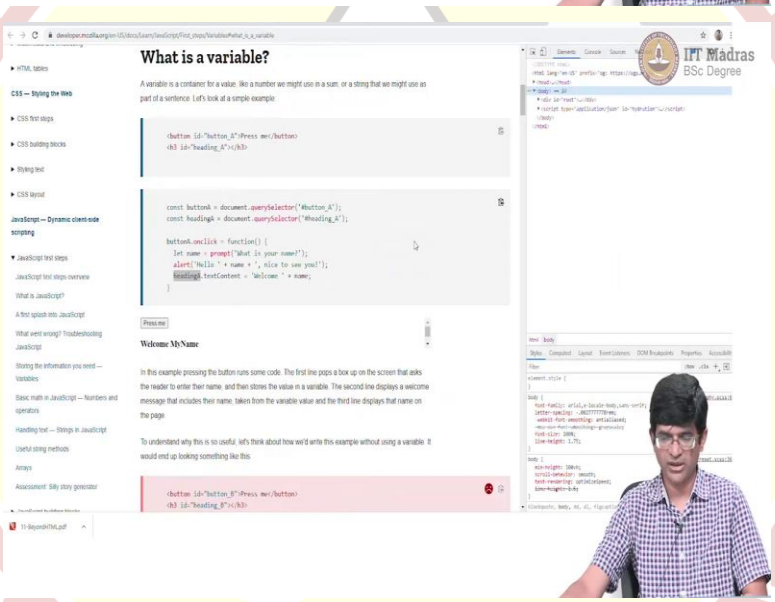


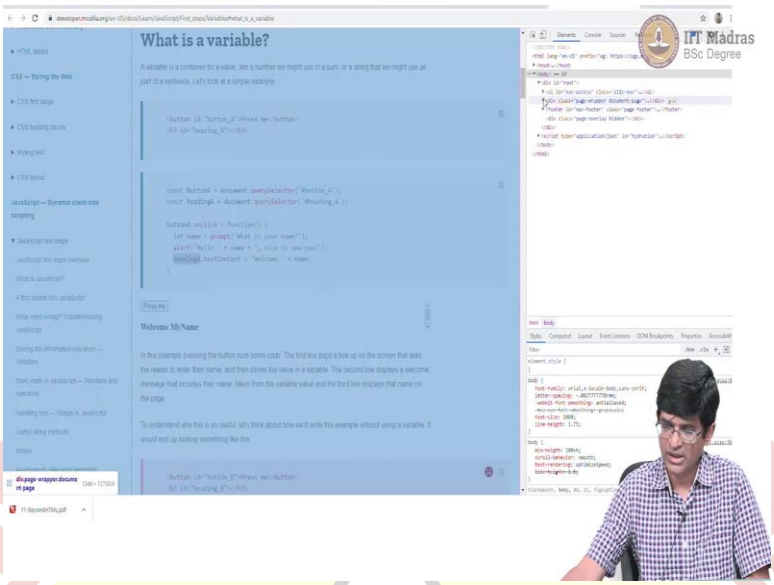


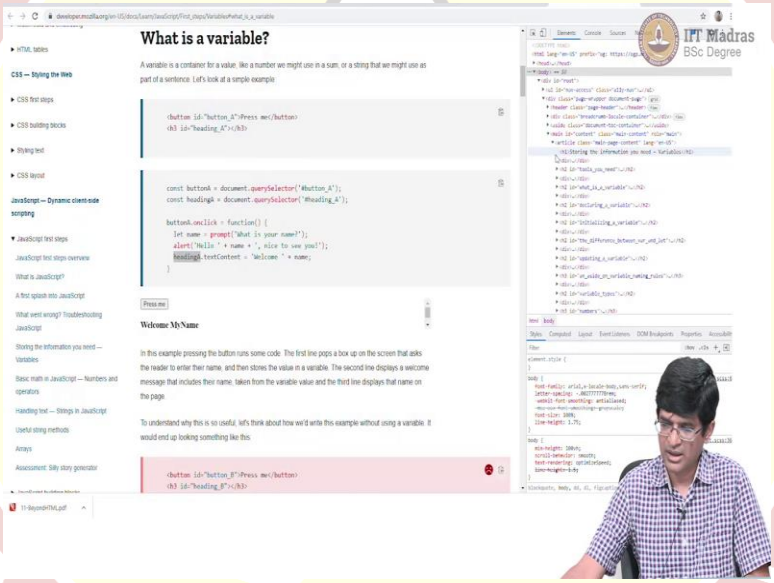

IIT Madras
BSc Degree
An embedded page at yari-demos.prod.mdn.mozit.cloud says
What is your name?
MyName
OK
Cancel
What
A variable is a c
tring that we might use as
part of a senten
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Press me
In this example pressing the button runs some code: The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
Multimedia and embedding
HTML tables
CSS — Styling the Web
CSS first steps
CSS building blocks
Styling text
CSS layout
JavaScript — Dynamic client-side scripting
JavaScript first steps
JavaScript first steps overview
What is JavaScript?
A first splash into JavaScript
What went wrong? Troubleshooting JavaScript
Storing the information you need — Variables
Basic math in JavaScript — Numbers and operators
Handling text — Strings in JavaScript
Useful string methods
Arrays
Assessment: Silly story generator
JavaScript building blocks

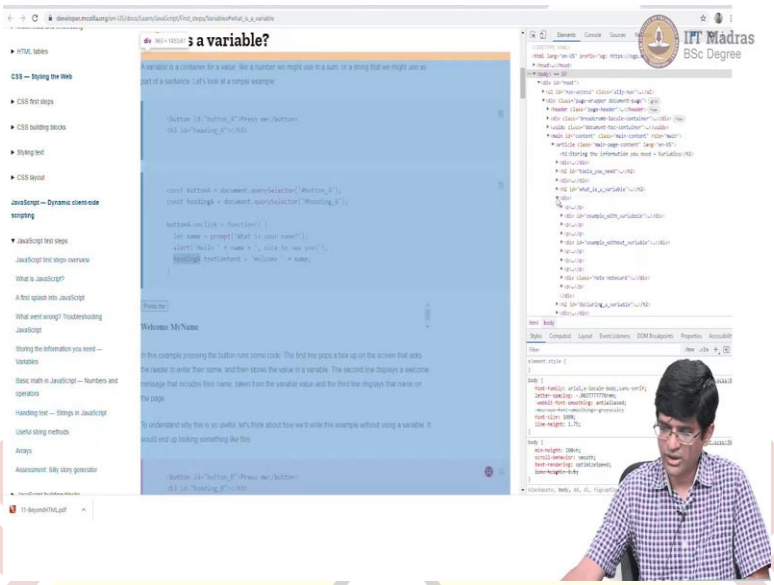

IIT Madras
BSc Degree
An embedded page at yari-demos.prod.mdn.mozit.cloud says
Hello MyName, nice to see you!
OK
What
A variable is a c
tring that we might use as
part of a senten
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Press me
In this example pressing the button runs some code: The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
Multimedia and embedding
HTML tables
CSS — Styling the Web
CSS first steps
CSS building blocks
Styling text
CSS layout
JavaScript — Dynamic client-side scripting
JavaScript first steps
JavaScript first steps overview
What is JavaScript?
A first splash into JavaScript
What went wrong? Troubleshooting JavaScript
Storing the information you need — Variables
Basic math in JavaScript — Numbers and operators
Handling text — Strings in JavaScript
Useful string methods
Arrays
Assessment: Silly story generator
JavaScript building blocks

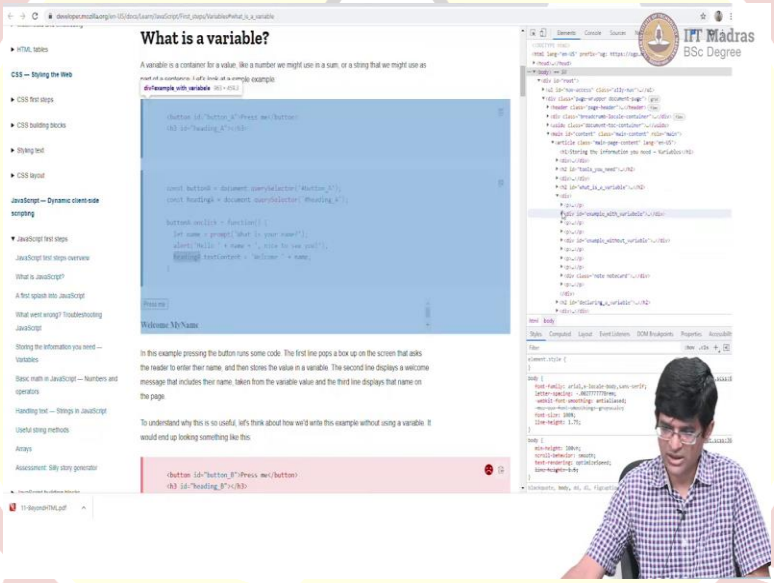

IIT Madras
BSc Degree
What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Press me
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>

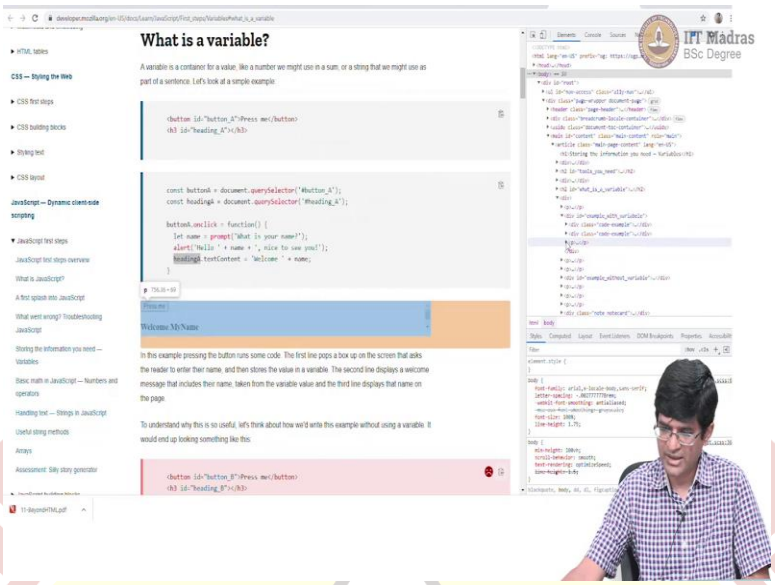

IIT Madras
BSc Degree
What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Press me
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>

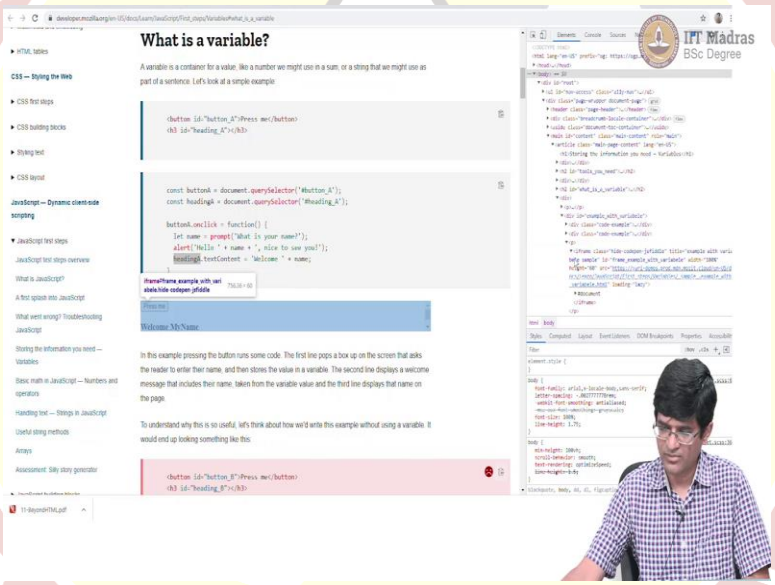

What is a variable?
IIT Madras
BSc Degree

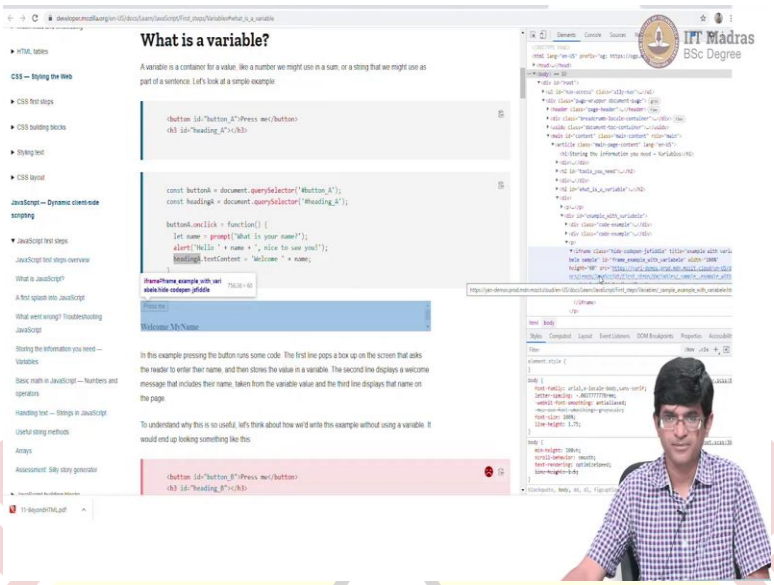

What is a variable?
IIT Madras
BSc Degree

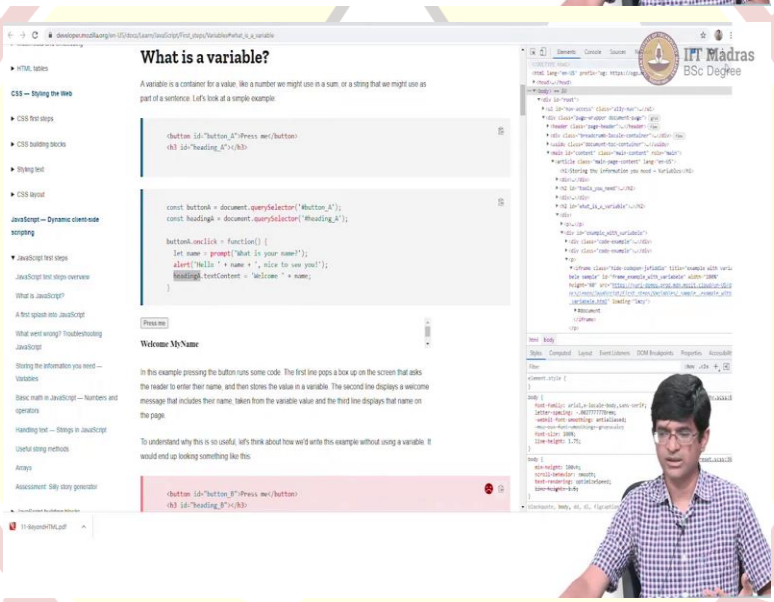

IIT Madras
BSc Degree

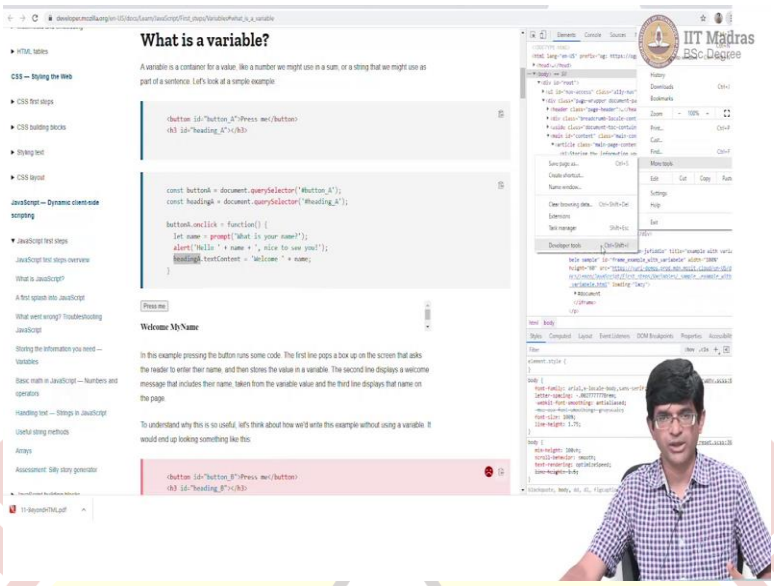

What is a variable?
IIT Madras
BSc Degree


What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
IIT Madras
BSc Degree


What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
IIT Madras
BSc Degree


What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
IIT Madras
BSc Degree


What is a variable?
A variable is a container for a value, like a number we might use in a sum, or a string that we might use as part of a sentence. Let's look at a simple example:
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');
buttonA.onclick = function() {
let name = prompt('What is your name?');
alert('Hello ' + name + ', nice to see you!');
headingA.textContent = 'Welcome ' + name;
}
Welcome MyName
In this example pressing the button runs some code. The first line pops a box up on the screen that asks the reader to enter their name, and then stores the value in a variable. The second line displays a welcome message that includes their name, taken from the variable value and the third line displays that name on the page.
To understand why this is so useful, let's think about how we'd write this example without using a variable. It would end up looking something like this:
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>
IIT Madras
BSc Degree



Now, a little bit more about JavaScript, a bit more of the basics. So for example, you have something which is a variable. So you have this concept of a variable in JavaScript. And as you can see over here, this is yet another example, what it says is I might want to have some kind of functionality which happens when I press this button.

How do I do that? I basically say that I will query the document to find out button A. Query the document to find out heading A. So both of these have been given some kind of names or IDs somewhere in the text. So you can see this <button id = button_A>, which is why when I query the document for hash button A, it finds this particular button.

Similarly, <h3 id =heading_A>, means that it will query and find that particular heading. And then I can say button A dot on click. And look at this. This is a slightly interesting way of doing things. I am straight away saying button a dot on click equals a function.

So, you remember what we said about JavaScript functions themselves being first class objects. I mean, I can basically take a function, assign it to a variable and so on. buttonA .onclick, onclick is an attribute of this button, which means you can think of it as a variable inside button. And I am creating a function without giving any name to that function, and just assigning that whatever I get out of the function over here, directly into onclick.

Now this, what I have done here, I have basically created a function without giving it a name, is something called an anonymous function, and is something that you will find a

lot of in JavaScript. It is one way of writing code. There was, the previous way that we showed where you explicitly wrote out a function and then did the assignment of the update is also perfectly fine, but this is a sort of clever and compact way of doing things and therefore you are likely to encounter this in several places.

What it does is basically once again it comes up with a prompt for the name, it does an alert and then afterwards it basically does, it is able to use this text content welcome plus name and directly create this. So the variable that you declared over here can be used.

Press this thing, and I just enter My Name, and click on OK. And the first thing it does is it runs the alert which says, hello My Name, nice to see you. And when I click OK, it now puts in the heading as well, headingA.textContent=welcome My Name.

So, all of these are basically examples of directly changing. The interesting thing over here is you will notice that you, you have actually gone and changed the html content of the page. There is no way of really differentiating this. The one thing that happens though is if you look at the page source you will not find My Name anywhere in it because that came about only after you modified the page.

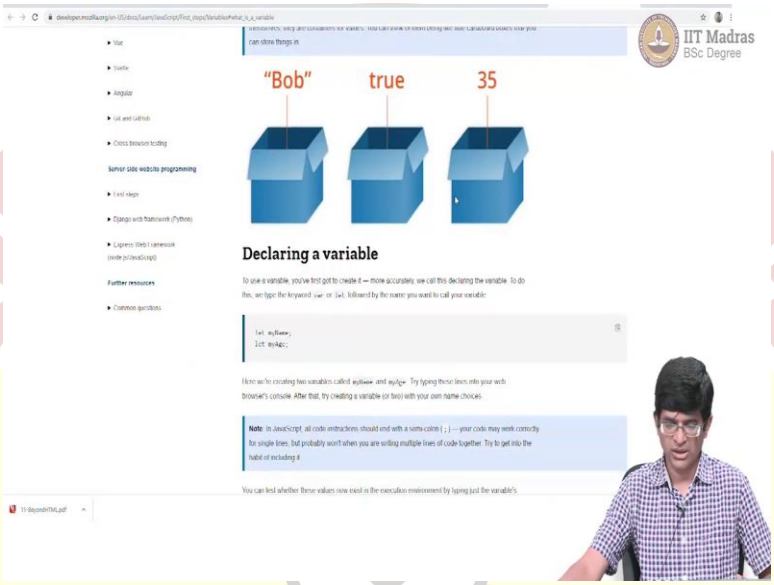
On the other hand, if you inspect using the sort of this F 12 button usually in Chrome, if you press that, then you will find that you can actually dive deeper into this. You can get into this part of it, where it sort of says I can go into the div, inside the page wrapper and inside the main article.

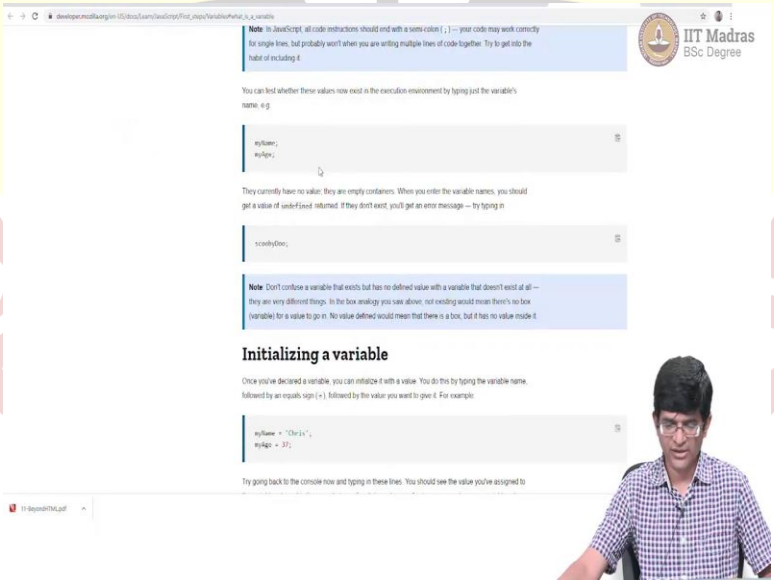
And I have this div over here which corresponds to this part over here, the example with the variable comes up over here, and as I look at the final output, I actually see that this corresponds to an I-frame, which, when I look inside it tells me whatever is actually running out here, and being displayed.

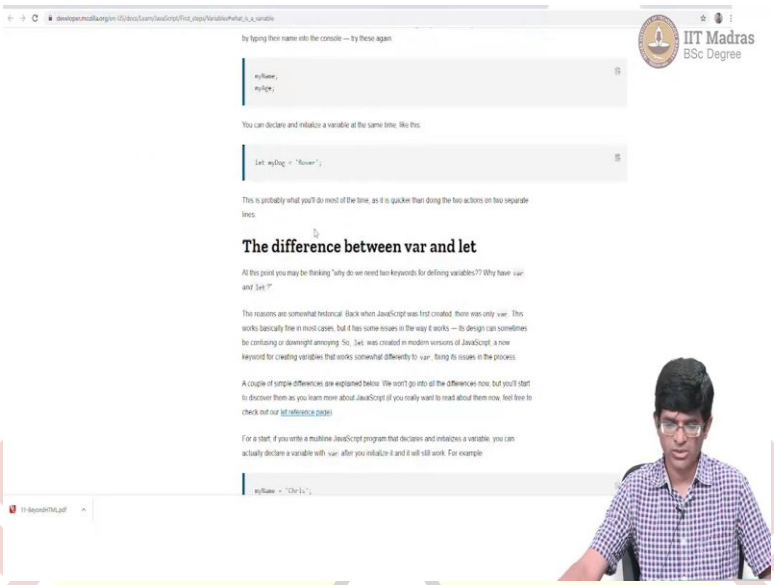
So, if you really want to find out which part of your page is being displayed according to what part of the code, you need to be able to use this webpage inspector. Once again, out of scope of this course but making use of it, if you basically go to the menu of any, of your browser usually, out here in the corner you would find that there are certain things like the more tools and you will find developer tools. That is how that thing comes up.

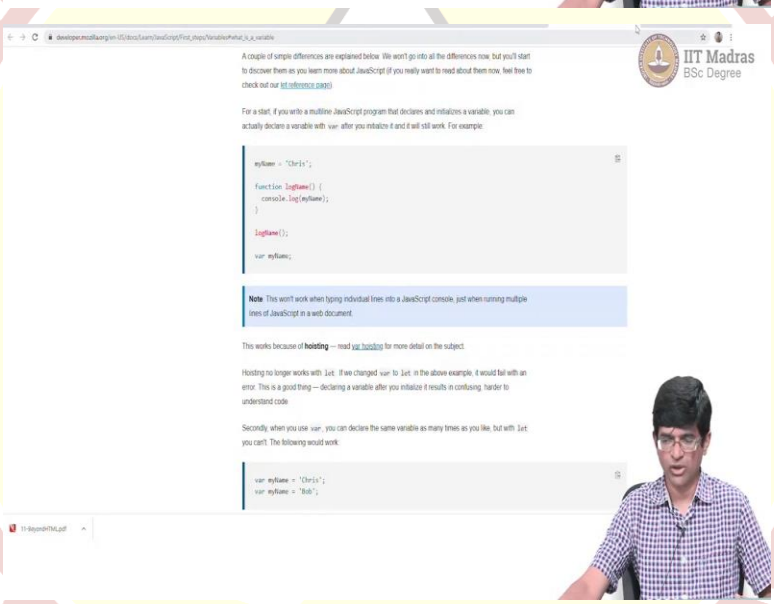
It is well worth playing around. It sort of shows you how different parts of the document are being displayed and so on. But you get into it when you are sort of getting into the details of how to actually design a page, how to have more control over different parts, what kind of JavaScript to write and so on.

(Refer Slide Time: 29:57)



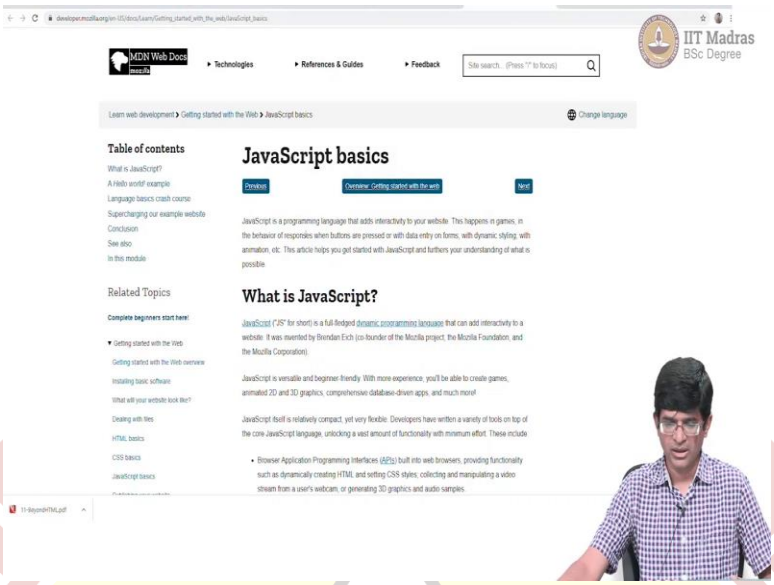


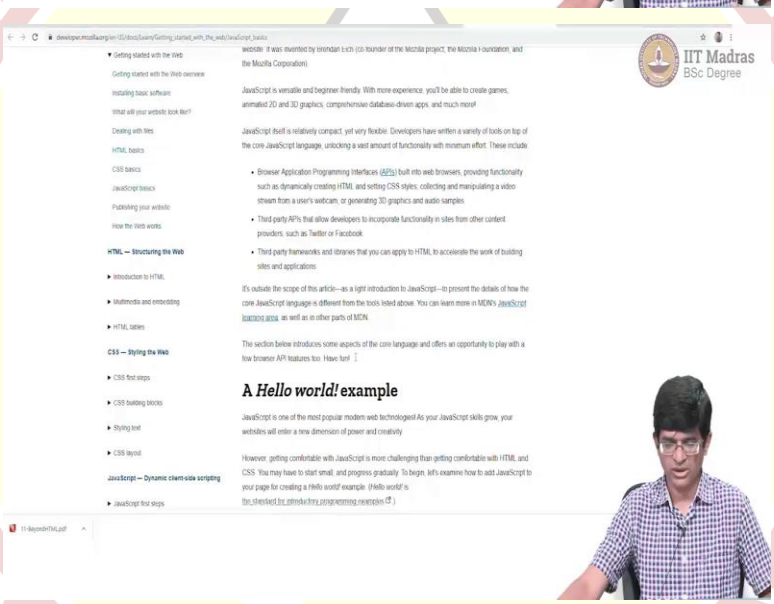


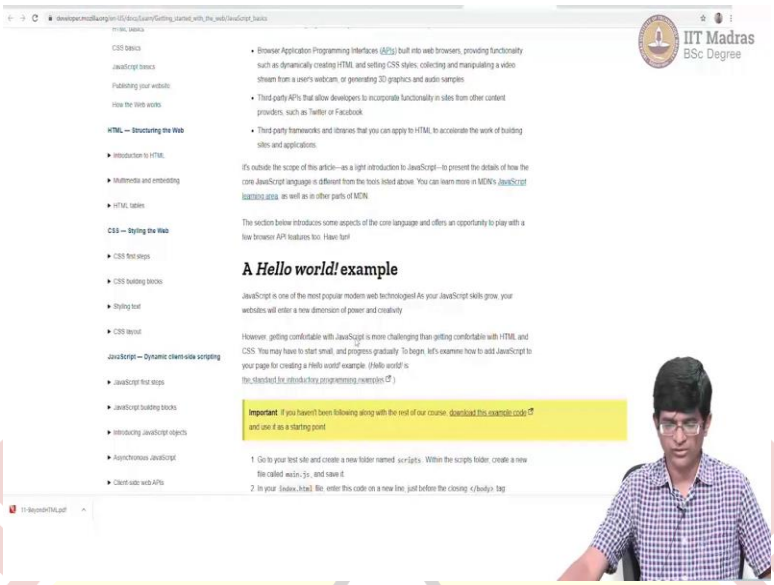


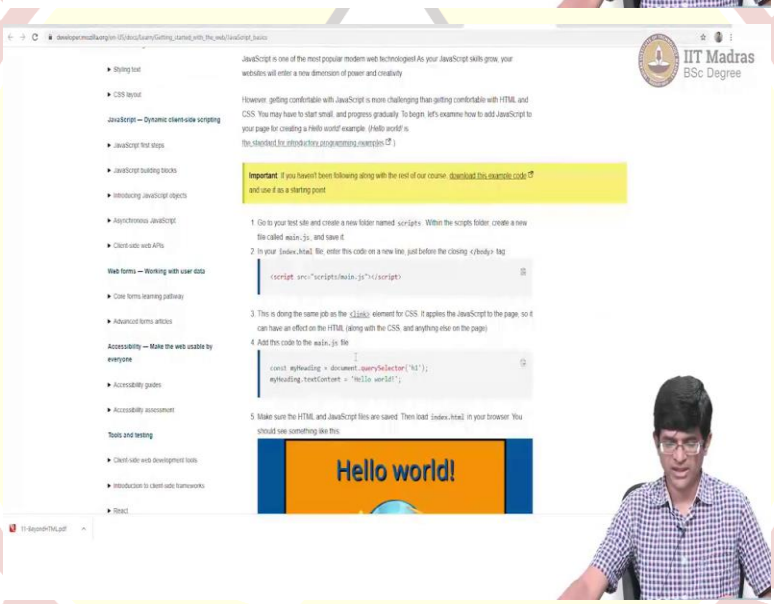
Similarly, there are many other ways of you what you can do with variables you can declare variables, you can initialize variables, there are some differences between var and let so on, which I am not getting into right now.

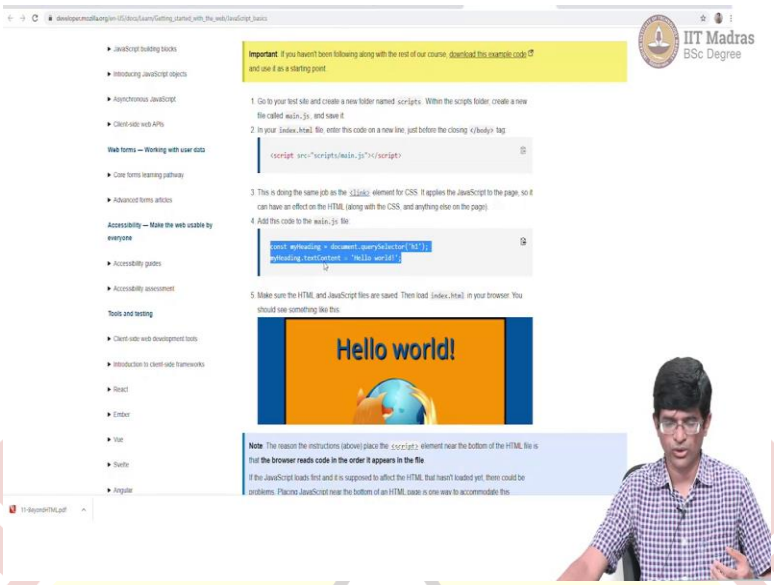
(Refer Slide Time: 30:06)

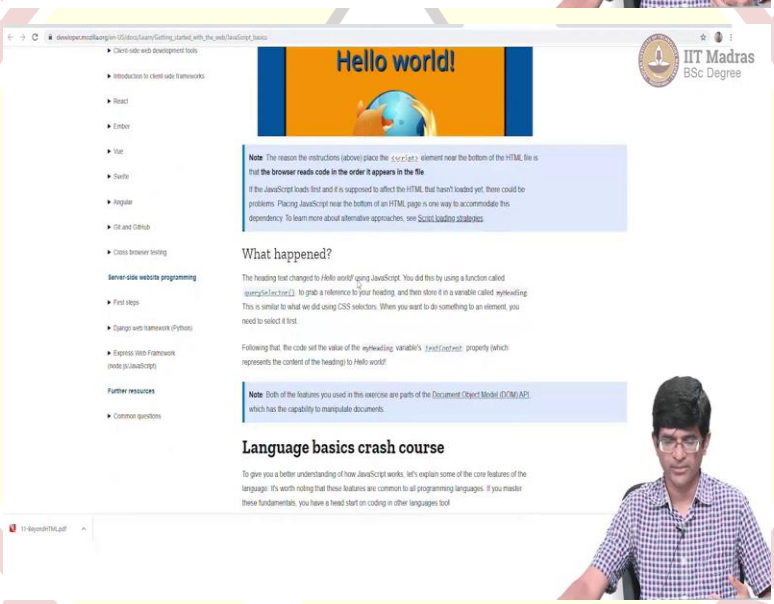

IIT Madras
BSc Degree
MDN Web Docs
Technologies
References & Guides
Feedback
Learn web development › Getting started with the Web › JavaScript basics
Change language
Table of contents
What is JavaScript?
A Hello world! example
Language basics crash course
Supercharging our example website
Conclusion
See also
In this module
Related Topics
JavaScript basics
Previous
Overview: Getting started with the web
Next
JavaScript is a programming language that adds interactivity to your website. This happens in games, in the behavior of responses when buttons are pressed or with data entry on forms, with dynamic styling, with animation, etc. This article helps you get started with JavaScript and furthers your understanding of what is possible.
What is JavaScript?
JavaScript ("JS" for short) is a full-fledged dynamic programming language that can add interactivity to a website. It was invented by Brendan Eich (co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation).
JavaScript is versatile and beginner-friendly. With more experience, you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!
JavaScript itself is relatively compact, yet very flexible. Developers have written a variety of tools on top of the core JavaScript language, unlocking a vast amount of functionality with minimum effort. These include
Browser Application Programming Interfaces (APIs) built into web browsers, providing functionality such as dynamically creating HTML and setting CSS styles; collecting and manipulating a video stream from a user's webcam, or generating 3D graphics and audio samples

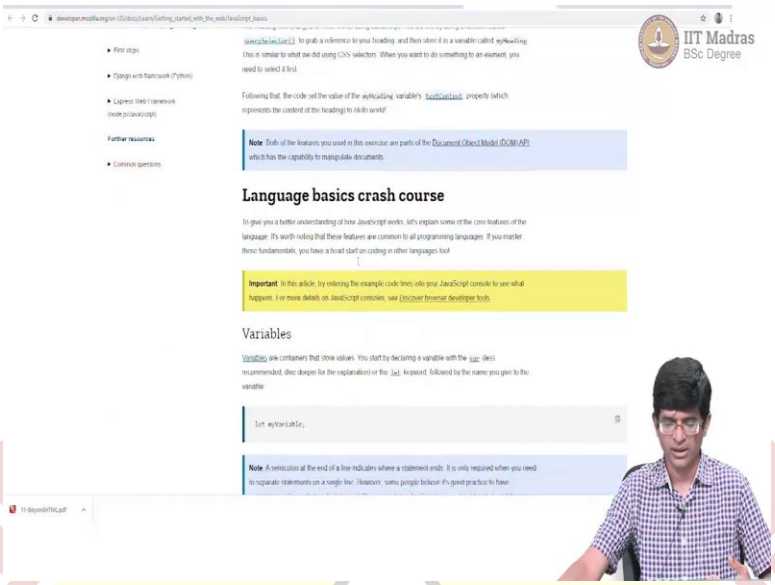

IIT Madras
BSc Degree
website. It was invented by Brendan Eich (co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation).
JavaScript is versatile and beginner-friendly. With more experience, you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!
JavaScript itself is relatively compact, yet very flexible. Developers have written a variety of tools on top of the core JavaScript language, unlocking a vast amount of functionality with minimum effort. These include
Browser Application Programming Interfaces (APIs) built into web browsers, providing functionality such as dynamically creating HTML and setting CSS styles; collecting and manipulating a video stream from a user's webcam, or generating 3D graphics and audio samples
Third-party APIs that allow developers to incorporate functionality in sites from other content providers, such as Twitter or Facebook
Third-party frameworks and libraries that you can apply to HTML to accelerate the work of building sites and applications
It's outside the scope of this article—as a light introduction to JavaScript—to present the details of how the core JavaScript language is different from the tools listed above. You can learn more in MDN's JavaScript learning area, as well as in other parts of MDN.
The section below introduces some aspects of the core language and offers an opportunity to play with a few browser API features too. Have fun!
A Hello world! example
JavaScript is one of the most popular modern web technologies! As your JavaScript skills grow, your websites will enter a new dimension of power and creativity.
However, getting comfortable with JavaScript is more challenging than getting comfortable with HTML and CSS. You may have to start small, and progress gradually. To begin, let's examine how to add JavaScript to your page for creating a Hello world! example. (Hello world! is the standard for introductory programming examples.)

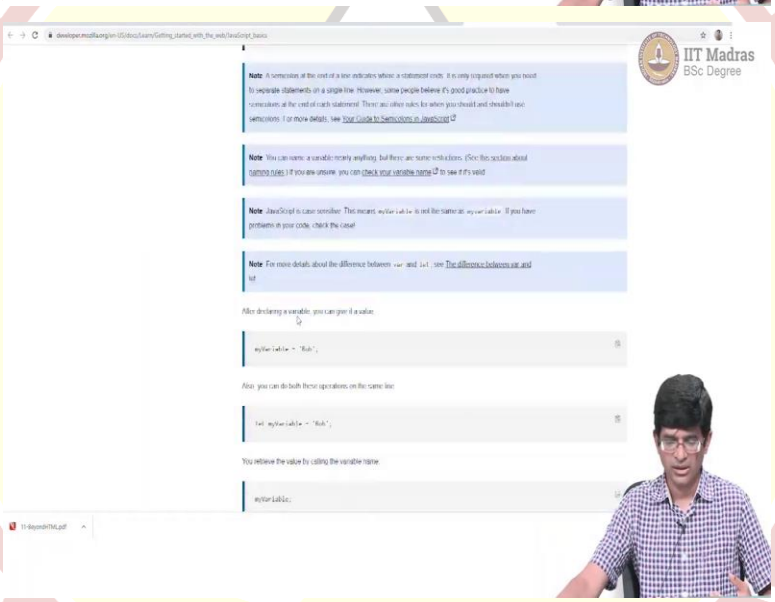

IIT Madras
BSc Degree
CSS basics
JavaScript basics
Publishing your website
How the Web works
HTML — Structuring the Web
Introduction to HTML
Multimedia and embedding
HTML tables
CSS — Styling the Web
CSS first steps
CSS building blocks
Styling text
CSS layout
JavaScript — Dynamic client-side scripting
JavaScript first steps
JavaScript building blocks
Introducing JavaScript objects
Asynchronous JavaScript
Client-side web APIs
Browser Application Programming Interfaces (APIs) built into web browsers, providing functionality such as dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from a user's webcam, or generating 3D graphics and audio samples
Third-party APIs that allow developers to incorporate functionality in sites from other content providers, such as Twitter or Facebook
Third-party frameworks and libraries that you can apply to HTML to accelerate the work of building sites and applications.
A Hello world! example
JavaScript is one of the most popular modern web technologies! As your JavaScript skills grow, your websites will enter a new dimension of power and creativity.
Important: If you haven't been following along with the rest of our course, download this example code and use it as a starting point.
1. Go to your test site and create a new folder named scripts. Within the scripts folder, create a new file called main.js, and save it.
2. In your index.html file, enter this code on a new line, just before the closing </body> tag


IIT Madras
BSc Degree
Styling text
CSS layout
JavaScript — Dynamic client-side scripting
JavaScript first steps
JavaScript building blocks
Introducing JavaScript objects
Asynchronous JavaScript
Client-side web APIs
Web forms — Working with user data
Core forms learning pathway
Advanced forms articles
Accessibility — Make the web usable by everyone
Accessibility guides
Accessibility assessment
Tools and testing
Client-side web development tools
Introduction to client-side frameworks
React
JavaScript is one of the most popular modern web technologies! As your JavaScript skills grow, your websites will enter a new dimension of power and creativity.
Important: If you haven't been following along with the rest of our course, download this example code and use it as a starting point.
1. Go to your test site and create a new folder named scripts. Within the scripts folder, create a new file called main.js, and save it.
2. In your index.html file, enter this code on a new line, just before the closing </body> tag
<script src="scripts/main.js"></script>
3. This is doing the same job as the <link> element for CSS: it applies the JavaScript to the page, so it can have an effect on the HTML (along with the CSS, and anything else on the page).
4. Add this code to the main.js file
const myHeading = document.querySelector('h1');
myHeading.textContent = 'Hello world!';
5. Make sure the HTML and JavaScript files are saved. Then load index.html in your browser. You should see something like this:
Hello world!


IIT Madras
BSc Degree
JavaScript building blocks
Introducing JavaScript objects
Asynchronous JavaScript
Client-side web APIs
Web forms — Working with user data
Core forms learning pathway
Advanced forms articles
Accessibility — Make the web usable by everyone
Accessibility guides
Accessibility assessment
Tools and testing
Client-side web development tools
Introduction to client-side frameworks
React
Ember
Vue
Svelte
Angular
Important: If you haven't been following along with the rest of our course, download this example code and use it as a starting point.
1. Go to your test site and create a new folder named scripts. Within the scripts folder, create a new file called main.js, and save it.
2. In your index.html file, enter this code on a new line, just before the closing </body> tag:
<script src="scripts/main.js"></script>
3. This is doing the same job as the <link> element for CSS. It applies the JavaScript to the page, so it can have an effect on the HTML (along with the CSS, and anything else on the page).
4. Add this code to the main.js file:
const myHeading = document.querySelector('h1');
myHeading.textContent = 'Hello world!';
5. Make sure the HTML and JavaScript files are saved. Then load index.html in your browser. You should see something like this:
Hello world!
Note: The reason the instructions (above) place the <script> element near the bottom of the HTML file is that the browser reads code in the order it appears in the file.
If the JavaScript loads first and it is supposed to affect the HTML that hasn't loaded yet, there could be problems. Placing JavaScript near the bottom of an HTML page is one way to accommodate this.


IIT Madras
BSc Degree
Client-side web development tools
Introduction to client-side frameworks
React
Ember
Vue
Svelte
Angular
Git and GitHub
Cross browser testing
Server-side website programming
First steps
Django web framework (Python)
Express Web Framework (node.js/JavaScript)
Further resources
Common questions
Hello world!
Note: The reason the instructions (above) place the <script> element near the bottom of the HTML file is that the browser reads code in the order it appears in the file.
If the JavaScript loads first and it is supposed to affect the HTML that hasn't loaded yet, there could be problems. Placing JavaScript near the bottom of an HTML page is one way to accommodate this dependency. To learn more about alternative approaches, see Script loading strategies.
What happened?
The heading text changed to Hello world! using JavaScript. You did this by using a function called querySelector() to grab a reference to your heading, and then store it in a variable called myHeading. This is similar to what we did using CSS selectors. When you want to do something to an element, you need to select it first.
Following that, the code set the value of the myHeading variable's textContent property (which represents the content of the heading) to Hello world!.
Note: Both of the features you used in this exercise are parts of the Document Object Model (DOM) API, which has the capability to manipulate documents.
Language basics crash course
To give you a better understanding of how JavaScript works, let's explain some of the core features of the language. It's worth noting that these features are common to all programming languages. If you master these fundamentals, you have a head start on coding in other languages too!





developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

IIT Madras
BSc Degree

Also, you can do both these operations on the same line:

```
let myVariable = 'Bob';
```

You retrieve the value by calling the variable name:

```
myVariable;
```

After assigning a value to a variable, you can change it later in the code:

```
let myVariable = 'Bob';
myVariable = 'Steve';
```

Note that variables may hold values that have different data types:

Variable	Explanation	Example
String	This is a sequence of text known as a string. To signify that the value is a string, enclose it in single quote marks.	`let myVariable = 'Bob';`
Number	This is a number. Numbers don't have quotes around them.	`let myVariable = 10;`
Boolean	This is a true/false value. The words `true` and `false` are special keywords that don't need quote marks.	`let myVariable = true;`
		`let myVariable = [1,'Bob','Steve',10];`

11-BeyondHTML.pdf

developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

IIT Madras
BSc Degree

Array	This is a structure that allows you to store multiple values in a single reference.	`let myVariable = [1,'Bob','Steve',10];` Refer to each member of the array like this: `myVariable[0]`, `myVariable[1]`, etc.
Object	This can be anything. Everything in JavaScript is an object and can be stored in a variable. Keep this in mind as you learn.	`let myVariable = document.querySelector('h1');` All of the above examples too.

So why do we need variables? Variables are necessary to do anything interesting in programming. If values couldn't change, then you couldn't do anything dynamic, like personalize a greeting message or change an image displayed in an image gallery.

Comments

Comments are snippets of text that can be added along with code. The browser ignores text marked as comments. You can write comments in JavaScript just as you can in CSS:

```
/*
Everything in between is a comment.
*/
```

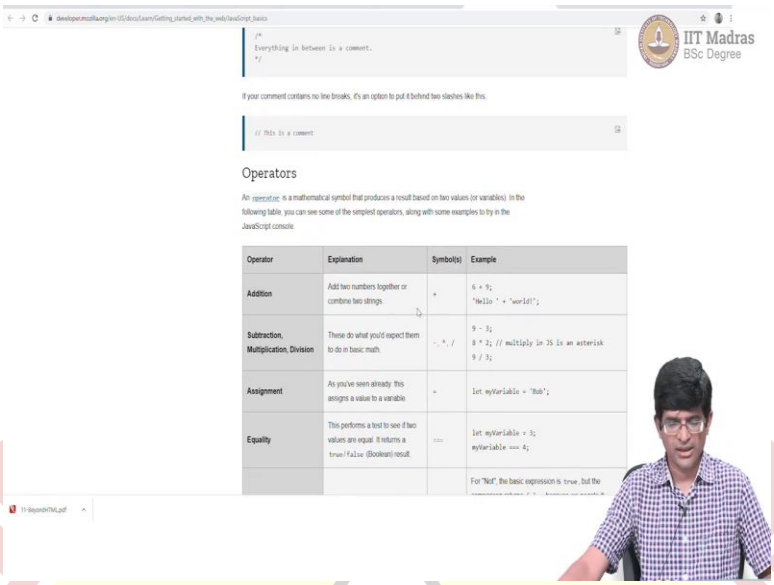
If your comment contains no line breaks, it's an option to put it behind two slashes like this:

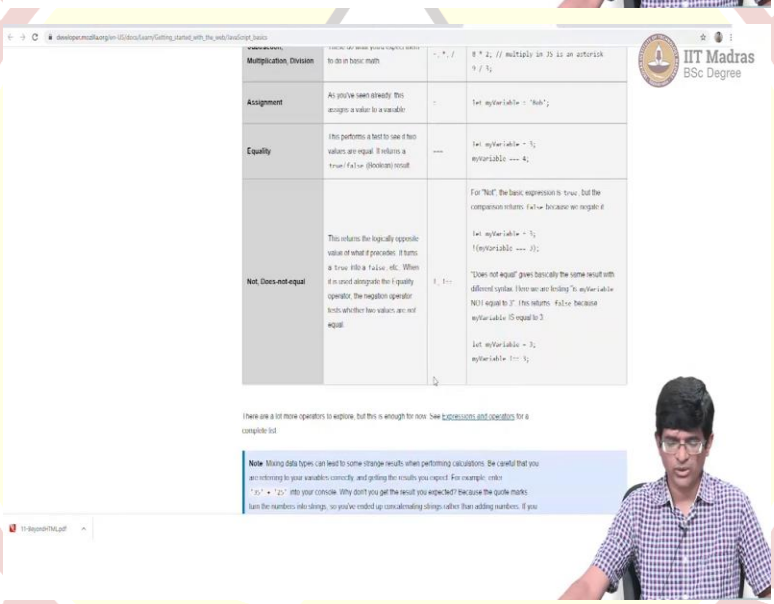
```
// This is a comment
```

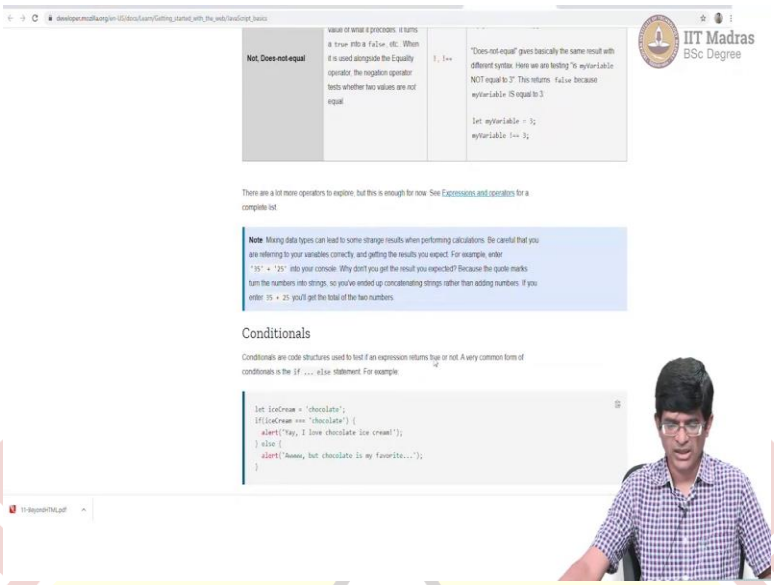
Operators

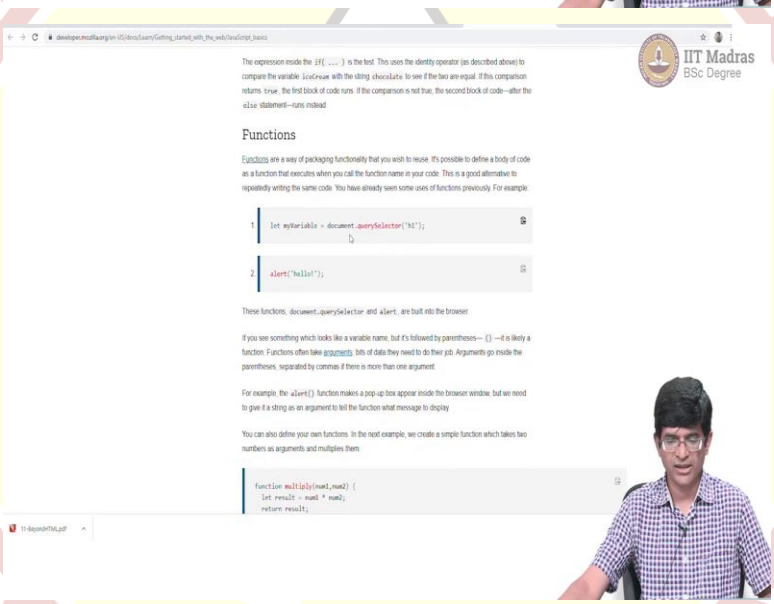
An `operator`, is a mathematical symbol that produces a result based on two values (or variables). In the following table, you can see some of the simplest operators, along with some examples to try in the JavaScript console.

11-BeyondHTML.pdf









developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

IIT Madras
BSc Degree

```
alert('hello!');
```

These functions, `document.querySelector` and `alert`, are built into the browser.

If you see something which looks like a variable name, but it's followed by parentheses— () —it is likely a function. Functions often take arguments: bits of data they need to do their job. Arguments go inside the parentheses, separated by commas if there is more than one argument.

For example, the `alert()` function makes a pop-up box appear inside the browser window, but we need to give it a string as an argument to tell the function what message to display.

You can also define your own functions. In the next example, we create a simple function which takes two numbers as arguments and multiplies them:

```
function multiply(num1,num2) {
  let result = num1 * num2;
  return result;
}
```

Try running this in the console, then test with several arguments. For example:

```
multiply(4, 7);
multiply(20, 20);
multiply(0.5, 3);
```

Note: The `return` statement tells the browser to return the `result` variable out of the function so it is available to use. This is necessary because variables defined inside functions are only available inside those functions. This is called variable scoping. (Read more about variable scoping.)

11-BeyondHTML.pdf

developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

IIT Madras
BSc Degree

to give it a string as an argument to tell the function what message to display.

You can also define your own functions. In the next example, we create a simple function which takes two numbers as arguments and multiplies them:

```
function multiply(num1,num2) {
  let result = num1 * num2;
  return result;
}
```

Try running this in the console, then test with several arguments. For example:

```
multiply(4, 7);
multiply(20, 20);
multiply(0.5, 3);
```

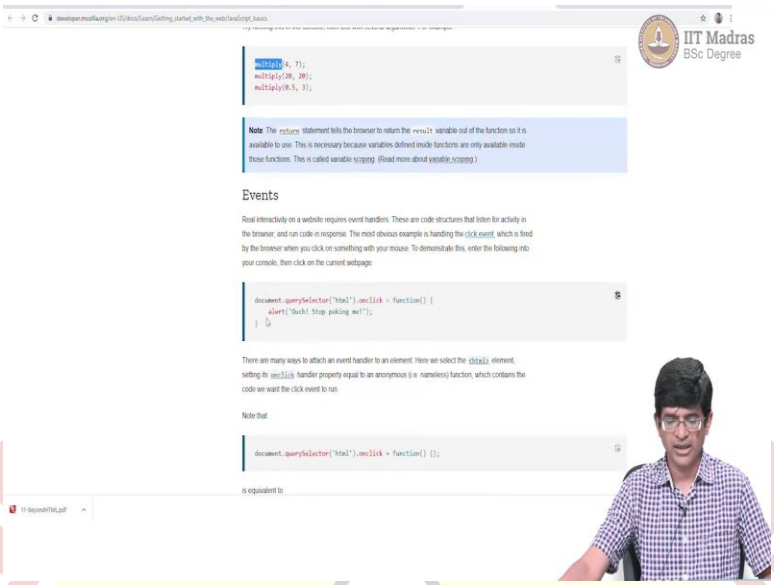
Note: The `return` statement tells the browser to return the `result` variable out of the function so it is available to use. This is necessary because variables defined inside functions are only available inside those functions. This is called variable scoping. (Read more about variable scoping.)

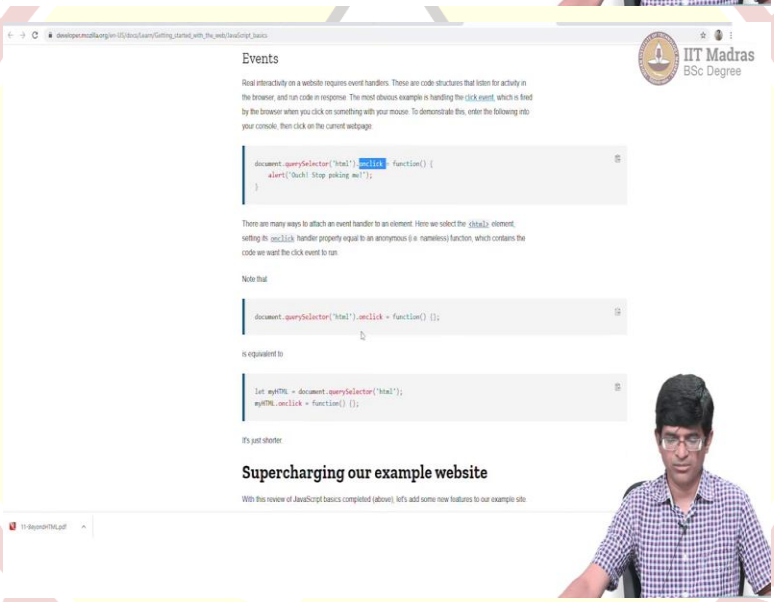
Events

Real interactivity on a website requires event handlers. These are code structures that listen for activity in the browser, and run code in response. The most obvious example is handling the click event, which is fired by the browser when you click on something with your mouse. To demonstrate this, enter the following into your console, then click on the current webpage:

```
document.querySelector('html').onclick = function() {
    alert('Ouch! Stop poking me!');
}
```

11-BeyondHTML.pdf





Instead, there is again on the same page, a set of an overview of the basics of JavaScript, which sort of tells you, first of all, how you can include a JavaScript into an html file, and inside the html file, you could have some extra information which basically says how do you write code, how do you declare a variable how do you assign a value to it.

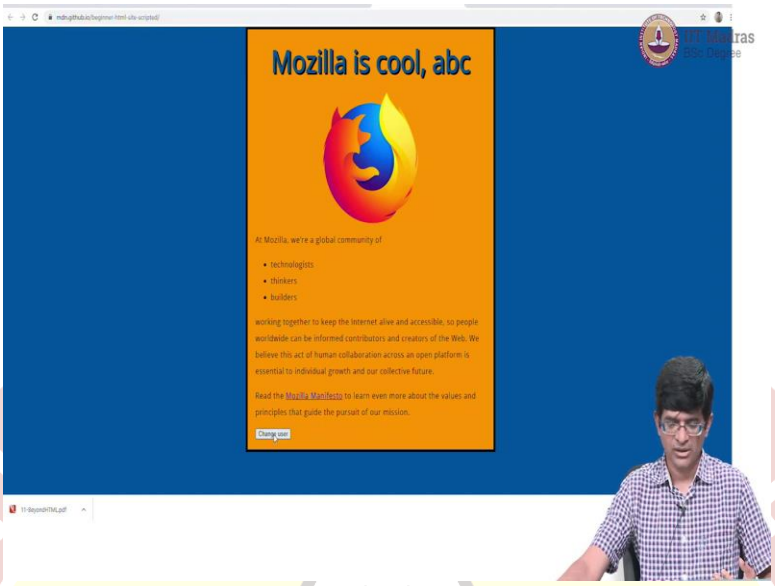
And as you go through you will find that there are a couple of things to keep in mind. Anything in /*, or //, this is basically exactly like what you would find in C, C plus plus, at least, or I think Java also. And this is how you define comments.

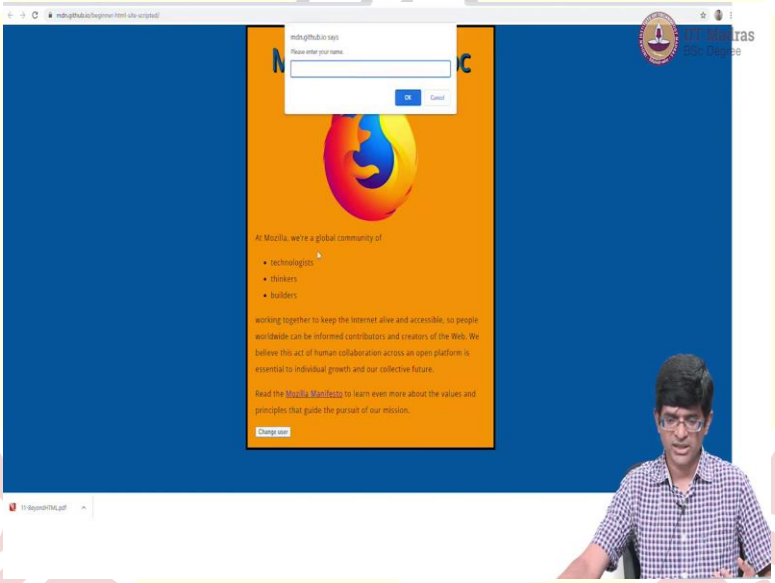
There are operators, +, -, *, etc just like you would find in pretty much any programming language. There are conditionals. You can have if some condition do something, else do

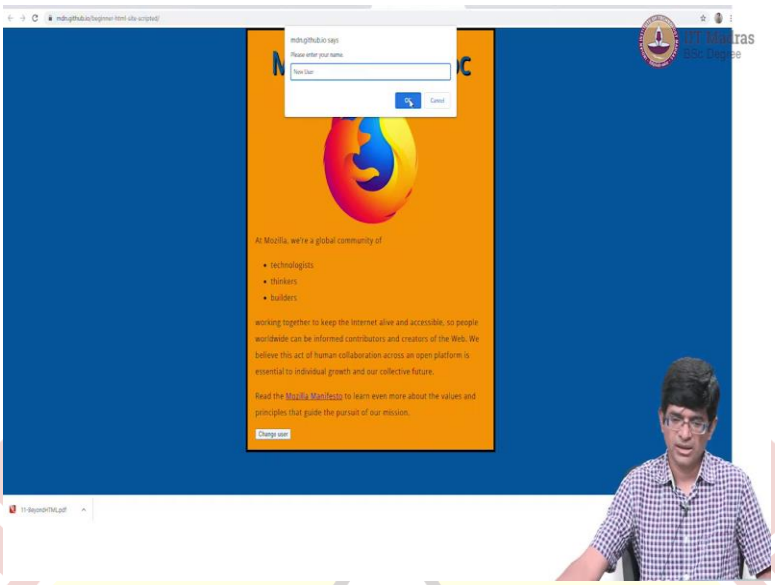
something else. And most importantly there are functions, which you can call, as we have already seen.

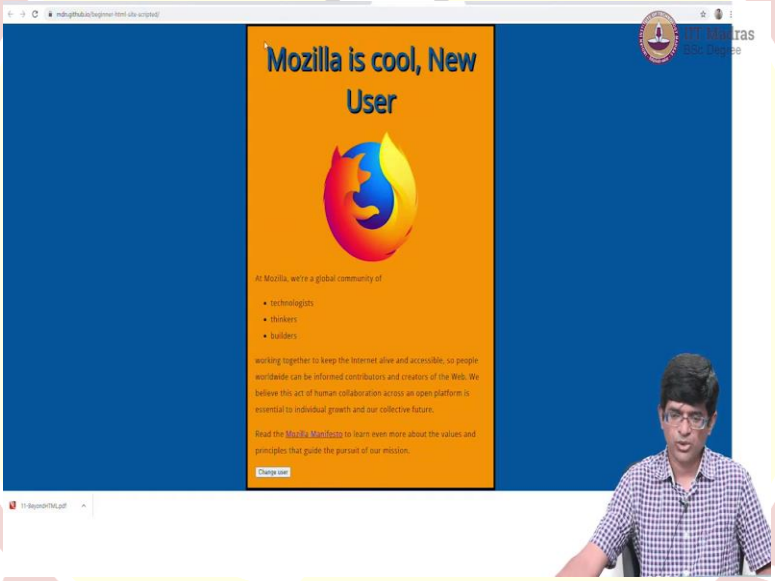
So, document.querySelector is a function, alert is a function you can define a function in this way similar to the def syntax in Python and then call the function by giving arguments. And there are events. So this whole on click, for example, can be used in order to update how the page responds to your inputs.

(Refer Slide Time: 31:40)



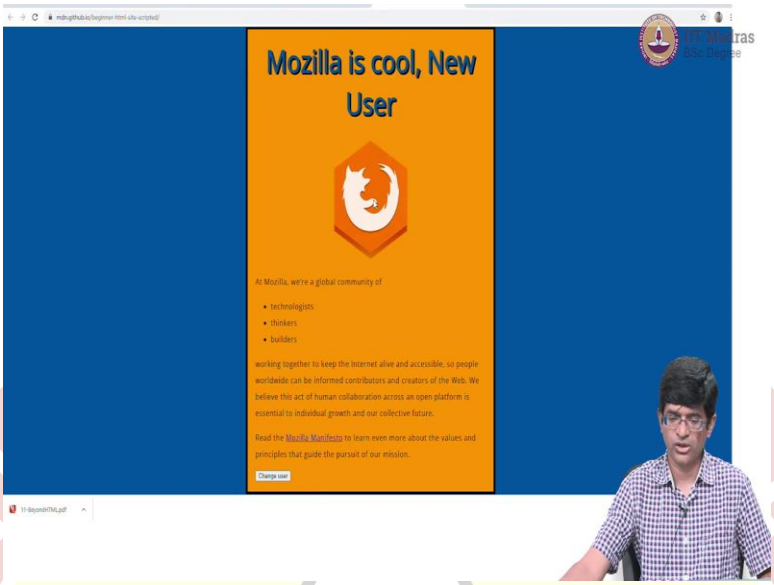






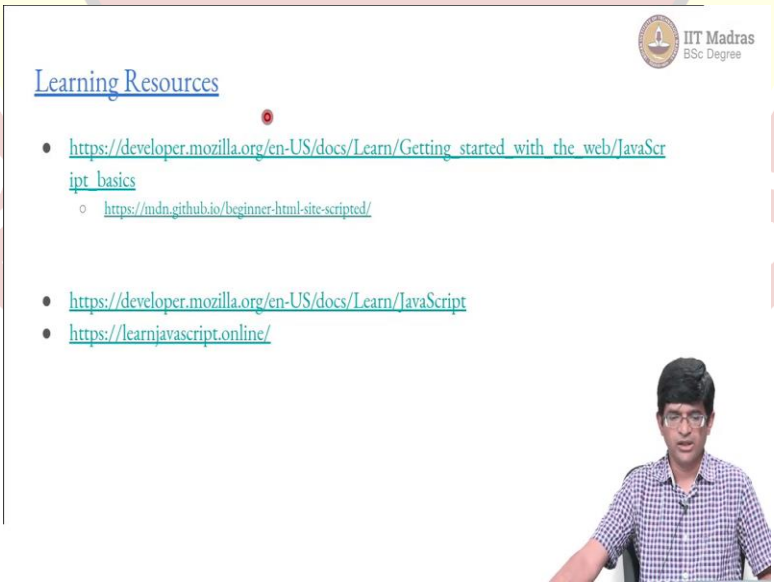
So, all of these are useful and if you look at this JavaScript basics they basically go about creating a page like this as a demo, and show you how each of the steps works. You can go and click on this change user, and say New User, whatever it is. At that point it says Mozilla is cool, New User.

(Refer Slide Time: 32:00)



And there is one more piece of JavaScript on this page which you might not even notice, which when you click on this image, it basically changes the image. So all that kind of functionality has now become like so smooth and fluid. It does not reload the page, nothing changes except the page itself as it is displayed, which means that the user interaction is a lot more smooth and clean.

(Refer Slide Time: 32:17)



So, there are a lot of learning resources for JavaScript. In particular, what I said about the Mozilla developer network, has a lot of useful information. They also recommend, there

is a learn JavaScript dot online, which is, the first few parts of it are free but then some parts of it are paid.

The important thing I would suggest is rather than going in and saying that I need to pay and take a course, the best way to learn any language including JavaScript is ultimately through example. Try programming, try something out on your own. That is the best way to really learn the language well.