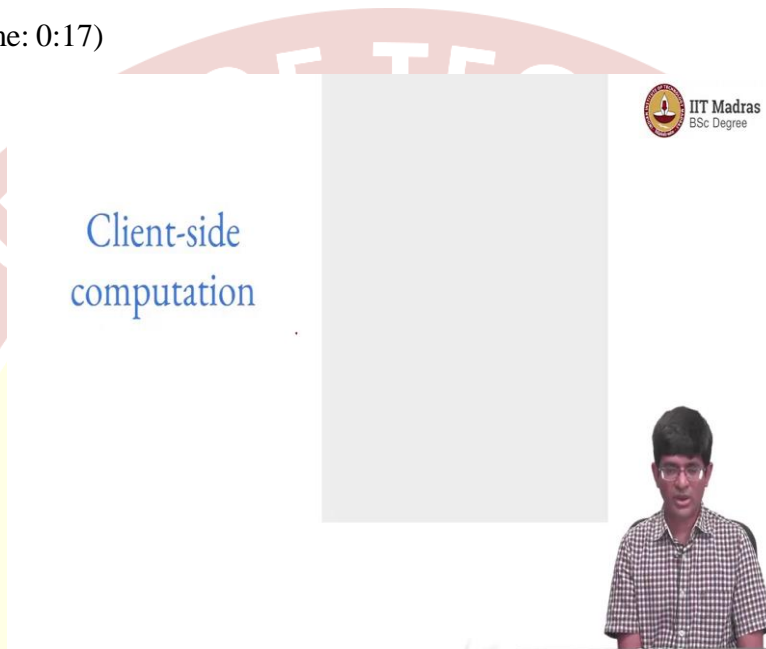


IIT Madras
ONLINE DEGREE

Modern Application Development – I
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Client Side Computations & Security Implications

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

(Refer Slide Time: 0:17)

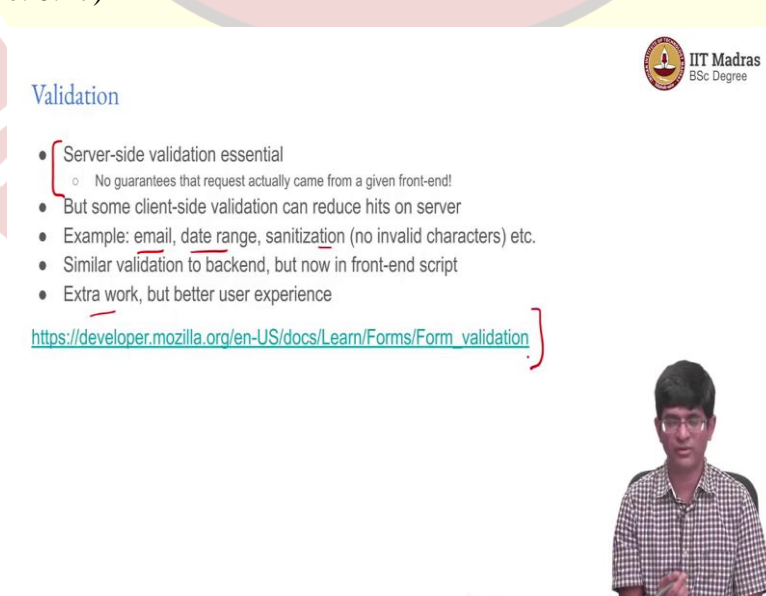


Client-side computation

IIT Madras
BSc Degree

So, now that we know different things that can, different ways in which a client can exist or can be implemented, let us take a further look at the notion of client side computation.

(Refer Slide Time: 0:27)



Validation

- Server-side validation essential
 - No guarantees that request actually came from a given front-end!
- But some client-side validation can reduce hits on server
- Example: email, date range, sanitization (no invalid characters) etc.
- Similar validation to backend, but now in front-end script
- Extra work, but better user experience

https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation

IIT Madras
BSc Degree

Now, one of the major forms of client-side computation that we need to perform is so called validation. Now before I go further into what validation looks like at the client side, I want to stress upon this fact, server-side validation, as we discussed in an earlier lecture is essential. Meaning that the last step before it actually hits the database has to have the request properly validated and to make sure that no invalid data can make it through into the SQL query or whatever, up to the database engine.

But if you do perform some amount of validation on the client side, it can actually help in reducing the number of hits on the server. Of course, if somebody wants to maliciously, try and cause problems for you that is a different story, they will just bypass your client altogether, and go directly to the server and start hitting it. That is not what we are talking about, we are talking about accidental mistakes or small problems here and there.

So, for example, there are many forms that require you to fill out an email, specify some kind of a date. And maybe even do some kind of basic sanitization on the data. Now, if you could do a lot of that on the Frontend itself, the big advantage that would happen is that you do not even have network traffic, you are not going to go and load the server unnecessarily when you have typed in something wrong.

There is some amount of extra work required, because now remember, you have to validate both at the Frontend and at the back end. But it results in overall a better user experience. Once again, in terms of details about this, this is a good link, the Mozilla Developer Network, they have a lot of information about form validation techniques, and so on.

(Refer Slide Time: 2:19)

Inbuilt HTML5 form controls

- Partial validation added by HTML5 standard
- `required`: mandatory field
- `minlength`, `maxlength`: for text fields
- `min`, `max`: for numeric values
- `type`: for some specific predefined types
- `pattern`: regular expression pattern match

Important: older browsers may not support all features.

Is backward compatibility essential for your app?



And one of the things that can be very powerful is to use the so-called inbuilt form controls that are there in HTML5. So, there are things where you can sort of specify that a particular, let us say, a name is required. You can also specify a minimum length and a maximum length for certain kinds of text fields. And the browser itself will validate that before passing it on to the server.

Remember, like I said, you might specify a maxlength over here. But you have to once again, check it at the server end, because there is no guarantee that the request that the server finally got, came through a particular client, or from a particular page. But like I said, this does help to cut down on sort of false alarms. You can do minlength maxlength for text fields, min max for numeric. Certain kinds of specific predefined types, which I am not very sure about.

There is also in fact, this something is quite powerful, something called regular expression pattern match. Now, what exactly are regular expressions, how do we use them, it is not something we will be getting into over here. But they can be used to do fairly complex pattern matching and sort of seeing if the input pattern matches with various kinds of things.

Now, one important thing to keep in mind is you might put all of this into your HTML page. Keep in mind that some older browsers may simply not support all the features. And one of the questions that as a developer, you then need to answer is, do you care? Is backward compatibility really required? In other words, are you concerned about something where a person might still be using, let us say, Windows 95 PC with the appropriate browser which was there on it? I mean, very unlikely. I mean, I do not think that even is going to be running at this point in time.

But you might have, let us say, some reasonably old version of Windows or Linux with a browser that does not support any of these capabilities. What do you do in such a case? Now, this is where frameworks help because frameworks very often take care of all of that behind the scenes. They sort of accommodate for the fact that you might be interested in backwards compatibility, and therefore put in extra code that takes care of all of that. Obvious problem is it also makes things much slower.

(Refer Slide Time: 4:42)



JavaScript validation

Constraint Validation API:

https://developer.mozilla.org/en-US/docs/Web/API/Constraint_validation

- Supported by most browsers
- Much more complex validation possible

Remember: not a substitute for server-side validation!



On the other hand, apart from the input validation that can be done as part of HTML5, you could also do validation with JavaScript. Remember that JavaScript has access to all your forms, all your elements. Therefore, they also provided something called the constraint validation API, which makes much more complex validation possible simply because you can now write a full-blown JavaScript function in order to do the validation.

(Refer Slide Time: 5:09)



```
<form>
  <label for="mail">I would like you to provide me with an e-mail address:</label>
  <input type="email" id="mail" name="mail">
  <button>Submit</button>
</form>
```



```
const email = document.getElementById("mail");  
  
email.addEventListener("input", function (event) {  
  if (email.validity.typeMismatch) {  
    email.setCustomValidity("I am expecting an e-mail address!");  
  } else {  
    email.setCustomValidity("");  
  }  
});
```



An example would be something like this, let us say that I have a form, where I essentially have one input type, where I have specified that the type is email. Now, what would that do, of course, I, even the fact that you specified it as email, you could basically do some basic validation within HTML5 itself or you could have a small piece of JavaScript code, which does this, it goes to the DOM, picks out the element by ID, which corresponds to mail, which is basically, this, id equal to mail out here.

And adds an event listener, which will listen for an input event on the id that is to say, you put something in there and press Enter or whatever to submit the form. And, at that point, if it basically does not satisfy the validity condition that you have, you would basically have in a message, which gives a custom message back to the end user, I am expecting an email address, not just that, wrong, or something of that sort.

(Refer Slide Time: 6:21)



```
const email = document.getElementById("mail");

email.addEventListener("input", function (event) {
  if (email.validity.typeMismatch) {
    email.setCustomValidity("I am expecting an e-mail address!");
  } else {
    email.setCustomValidity("");
  }
});
```

I would like you to provide me with an e-mail address:

 I am expecting an e-mail address!



And this is what it would look like. So, if I typed in something like this into such a form, and tried clicking on submit, it would pop up with a message saying, I am expecting an email address, whereas the normal default, if I had not used this custom validity would have been something different, whatever the browser just is going to say or it might even have just, not shown any message, but just refuse to accept that input. So, JavaScript obviously also helps you to add to what you can do with validation.

(Refer Slide Time: 6:48)

Captcha

- Problem: scripts that try to automate web-pages
- Can generate large number of requests in short time - server load
- Railway Tatkal, CoWin appointments etc.

Solution

- Prove that you are a human
- Limited number of clicks possible per unit time
- Script on page will generate some token - server will reject requests without the token



Now, validation is not the only thing that you can do with client-side computation. You can also do things like captcha. And why do we need something like captcha, the basic problem

that is trying to solve is that whenever you have a web page, somebody will probably try to cause problems for you.

So, they try to automate write a script, which will try to automatically do some things or hit the page multiple times, and so on. Examples are, the IRCTC, for railway Tatkal bookings, they have this problem where people try to write scripts, which will automatically bypass everything and go through and book it before someone else can.

Similarly, we also had a lot of reports of applications around the time that the CoWin vaccine app came out saying that, there were ways by which it would allow you to just do everything very fast and get the application and get the appointment before it, before others could do it. So obviously, this is happening because of scripts. We want to sort of prevent that from happening and prove that you are a human in order to be able to access a given webpage.

And what are the restrictions of humans? At the end of the day, when I am using a mouse and clicking there are limits on how many clicks and how much data I can enter within a given amount of time. And what things like re-captcha from Google do, is that it basically runs a script on the page, which is sort of tracking what you are doing with the mouse, where you are going, what you are clicking on, what you are entering and so on, and sort of builds up information about whether or not you are a human.

Which is why in many cases, you might notice that, the re-captcha basically gives you a box, it does not ask you to type anything in, you just click on the box, say I am a human, and it says, accept it and move on. Now, the reason it is doing that is it has already been tracking information about what you have been doing with the mouse what you have been doing with keys and so on. And therefore, it has reasonable confidence that yes, you are actually human who is using this.

And if it has a doubt, of course, you know it, you put up some pictures of traffic lights, or dogs or cats and ask you to select them so that you can then continue to the next phase. So that is one potential use for JavaScript, the fact that it can track a lot of information. But potentially, that is also concerning. The fact that the script is able to keep track of so much about you enough to decide whether or not you are a human, should be a little bit worrying. But anyway, we are used to it now.

(Refer Slide Time: 9:23)



Crypto-mining ?

- Javascript is a "complete" language
- Can implement any computation with Javascript
- Modern JS engines very powerful, fast
 - Can even access system graphics processor (GPU) for rendering etc.
- Run a simple page that loads and runs a JS script
- Script will send results back to server through async calls
- Client may not even be aware!



What else can you do with client-side computations? Pretty much anything and in fact, there are cases where people have even written scripts that do crypto mining. So, Bitcoin things of that sort, try and run it. Now, what is cool about this is that basically now, all that you need to do is put it up on your web page, and anybody visiting that web page is now going to start running a crypto mining algorithm on their machine. It is not on your server.

Now, obviously, this is frowned upon and whenever this is seen people take active steps to make sure that those scripts are taken out. So, it is definitely not advisable it can get you seriously blacklisted and in a lot of trouble. It is considered at the same level as trying to sort of crack into servers. So, it is to be avoided. But the point is that you can do lots of interesting things.

And, of course, the reason why all this happens is that modern JavaScript engines are really powerful, they are able almost to run at the full speed of the underlying processor, which is very, very fast. And in a lot of cases, they can even access the system GPUs and thereby get access to really high-performance computation. And what happens in such a case is that, there is a simple page that loads runs a JavaScript script.

And that script will, in turn, do some computation and make calls to an external server in order to send data back through asynchronous calls in the background. The client may not even be aware that all this is running. So, the point is, all of this is also something that can be done as part of client-side computation, not necessarily a good thing, but it is possible.

(Refer Slide Time: 11:01)

Security Implications



So finally, we come to the security implications of all of this, so you have a full-blown language now running in your browser Frontend, what does that mean?

(Refer Slide Time: 11:13)

Sandboxing



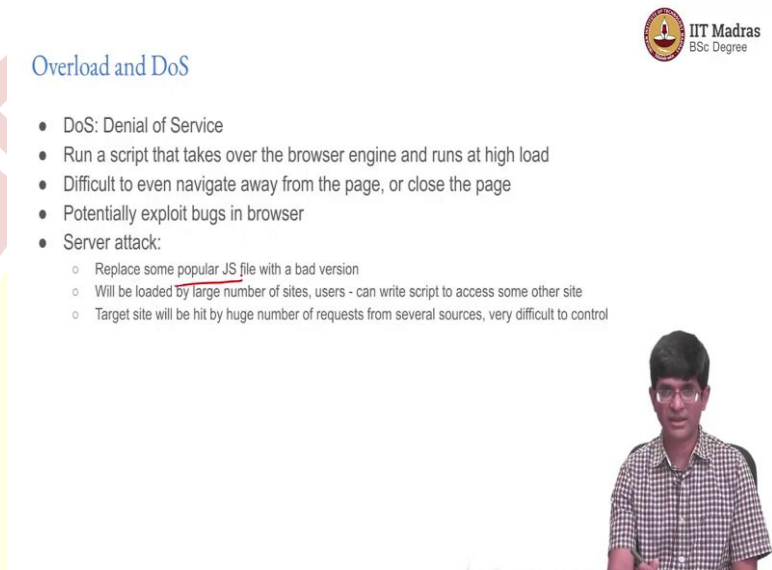
- Should JS be run automatically on every page?
 - Yes: provides significant capabilities
 - No: what if the page tries to load local files and send them out to server?
- Sandbox: secure area that JS engine has access to
- Cannot access files, network resources, local storage
- Similar to a Virtual Machine, but at higher level (JS interpreter)



That is a big question to be asked, should JavaScript be run automatically on every page? And if you answer yes, great, because it provides a lot of additional capabilities and can make your user experience a lot better and in many ways. But you might also say no, simply because, what if that page now tries to load local files and somehow send them out to a server, which does not have any business seeing those local files, I do not want my documents to be just sent out to someone else.

So, there, a concept called sandboxing is used, which basically provides a secure area that the JavaScript engine has access to, the JavaScript engine is limited to the sandbox. It cannot access anything outside of that, in particular, it cannot access files, it cannot access network resources, local storage, nothing of that sort. Because all of those are outside the sandbox. So, it is similar to a sort of restricted virtual machine, but at a higher level, it is basically run in the JavaScript interpreter and not at the sort of machine instruction level.

(Refer Slide Time: 12:20)



Overload and DoS

- DoS: Denial of Service
- Run a script that takes over the browser engine and runs at high load
- Difficult to even navigate away from the page, or close the page
- Potentially exploit bugs in browser
- Server attack:
 - Replace some popular JS file with a bad version
 - Will be loaded by large number of sites, users - can write script to access some other site
 - Target site will be hit by huge number of requests from several sources, very difficult to control

Now, that still means that there are ways in which you can cause problems for the client. And, or not just for the client, even for certain servers. One of the things that has been done in the past is potentially, take a simple, some kind of a popular JavaScript file somewhere and replace it with a bad version.

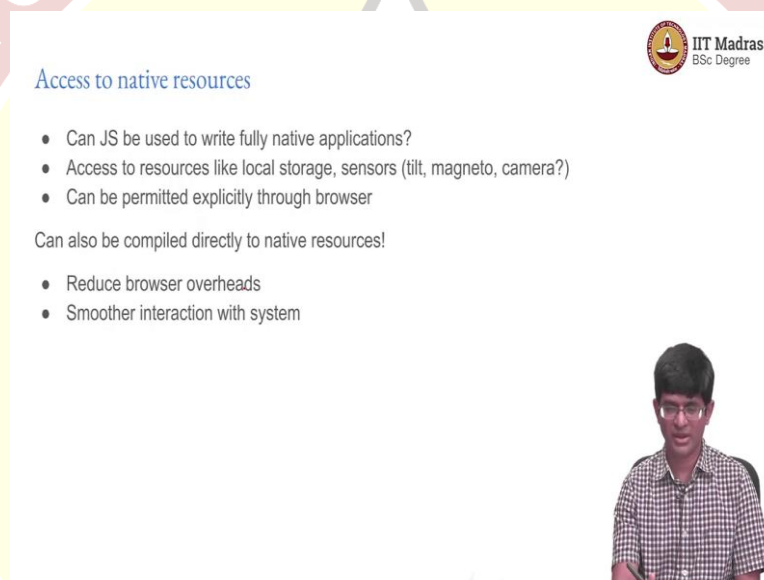
Now, the problem with this bad version is that maybe, it even behaves normally most of the time, but under some specific conditions, it will start generating requests to one central server. So now just imagine that this is a popular JavaScript file, it is being loaded onto the web pages of thousands, or even millions of people around the world.

And at some specific point in time, all of those machines, as long as the person is browsing a page that has that JavaScript loaded, suddenly start generating requests to apple.com, or some website. That server is most likely going to go down under what is called a denial-of-service attack. It is just hit by so many requests that it has to stop responding to any of them and wait for whatever, this entire thing to subside.

So, denial of service is potentially possible, you could also do the opposite, which is that you might run something which runs on the client, and uses up so much resources, so the client that it is difficult to even navigate away from that page or close the browser. Once again, the clients machine becomes unusable. So, all of those are possible, JavaScript engines typically have a lot of safeguards in them to prevent things of that sort from happening.

So, it is more likely that you will get a message saying, look, this page is using too many resources, should we kill it? Rather than sort of just hanging the entire system. But there might be ways, especially if there are bugs or something that are unknown to the user, that can actually cause problems.

(Refer Slide Time: 14:22)



The slide is titled "Access to native resources" and features the IIT Madras BSc Degree logo in the top right corner. It contains a bulleted list of questions and a statement about native resources, followed by another bulleted list of goals. A small video inset in the bottom right shows a man speaking.

Access to native resources

- Can JS be used to write fully native applications?
- Access to resources like local storage, sensors (tilt, magneto, camera?)
- Can be permitted explicitly through browser

Can also be compiled directly to native resources!

- Reduce browser overheads
- Smoother interaction with system

And the other big question, as far as security is concerned, is of course, access to native resources. Should a JavaScript application be allowed to access native resources? Can it be used in order to write native like applications? If you can allow access to local sensors, for example, the tilt sensor, cameras, magnetometers and so on or local storage, you can actually create very nice applications that are completely web based and run in your browser.

And in a lot of cases, what happens usually for something like this is that it has to be explicitly permitted by the user. And then it will be sort of installed locally onto your system in such a way that, the user is aware of the fact that yes, this is a locally installed thing that is running, it is, you have to trust it at some level. But that is, you are trusting it just about as much as any program that you download from the internet and run.

It is also possible to sort of compile directly down into native resources, which means that it will use the APIs of the underlying operating system. In such a case, you can have even smoother interaction with the system and potentially, better, I mean, reduce browser overheads even further.

(Refer Slide Time: 15:42)



 IIT Madras
BSc Degree

Summary

- Frontend experience determined by browser capabilities
 - Basic HTML + CSS rendering - styling
 - Javascript / client-side scripting for user interaction, smoother integration
- Native clients possible
- Potentially serious security implications!
- Always validate data again at server, do not assume client validation
 - HTTP is stateless: server cannot assume client was in a particular state!

So, to summarize all of this talk about the Frontend and JavaScript, the Frontend experience is determined to a large extent by browser capabilities. The basic HTML plus CSS based rendering, the CSS takes care of styling, the HTML takes care of the content. That can be enhanced quite significantly by having some level of scripting, which is usually provided today by JavaScript.

So, JavaScript or client-side scripting can make for a much better user experience. And also open up possibilities that might be very difficult to do with only pure HTML and CSS, for the simple reason that JavaScript is a full-blown programming language sitting on the client side. Potentially very serious security implications that you need to be aware of as a developer.

And, of course, the last thing that bears repetition, so I am going to say it again and again, is no matter what validation you might do in your JavaScript or at the Frontend, you always have to make sure you are doing it once again at the Backend.

The server, in other words, should any time a server is written using web technologies, it means that it can receive an HTTP request that could potentially be from anywhere, which means that it cannot assume anything about the state of the client or the nature of the client,

and has to be ready to do the validation once again on its own before it allows the data or the request to go through into the database.

