

IIT Madras
ONLINE DEGREE

Modern Application Development II
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Prof. Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology- Madras

JavaScript Origins and Overview

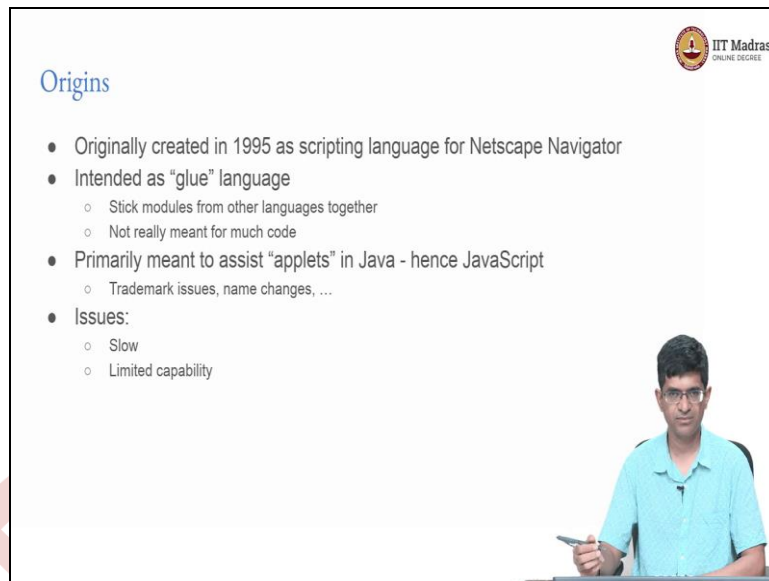
(Refer Slide Time: 00:17)



Hello everyone, welcome to modern application development part 2. Now since this course is going to focus a lot in a large part around javascript and its uses. It is useful to spend a little time trying to understand a bit of the history and the origins of javascript in particular why and how it originated has implications on how it works and it is sort of good to at least understand some of those.

So, that we are prepared for it and we either know how to work around it or you know how to use more modern versions of the language that sort of get around some of those restrictions.

(Refer Slide Time: 00:52)



Origins

- Originally created in 1995 as scripting language for Netscape Navigator
- Intended as "glue" language
 - Stick modules from other languages together
 - Not really meant for much code
- Primarily meant to assist "applets" in Java - hence JavaScript
 - Trademark issues, name changes, ...
- Issues:
 - Slow
 - Limited capability

So, where did javascript originate from. So, somewhere around 1995 or a little bit before that is when sun microsystems came up with this programming language called java and java was essentially designed in a way that it could be used in order to implement something called applets. What that meant was an applet would be something that would run within a web browser and would allow tremendous interactivity it would basically allow you to have a full blown sort of native application running in the browser.

Now what that meant was it had to be sort of defined as something which would run its in its own virtual machine in order to minimize the effects of security issues we do not want some kind of an application that is just downloaded from a web page running on an app on or running on anybody's computer because it could then read their files it could send sensitive information somewhere else and so on.

So, java was created with a lot of restrictions around how it could be used but it also stood out very clearly from the rest of the web page because you had to sort of define one area in the page where the plugin would run the java plugin. And the java applet would run within that area and within that area all the interaction would be different from the rest of the page.

Even how the cursors looked what happens when you click on a certain space all of that would. Now be handled by the java applet and not by the web browser at all. Now that is powerful but also sort of separates out completely between the browser and the java applet right. So, Netscape Navigator which was one of the most popular browsers at that

time and which has eventually evolved into what we now know today as Mozilla Firefox was one of the best browsers at the time.

And one of the people working there created a scripting language for Netscape Navigator in 1995. Now a scripting language of course as you know because you are familiar with python. The primary advantage of it is simplicity it is easy to use a scripting language you can just sort of get started you write the language and you know you run it through the interpreter and you are good to go.

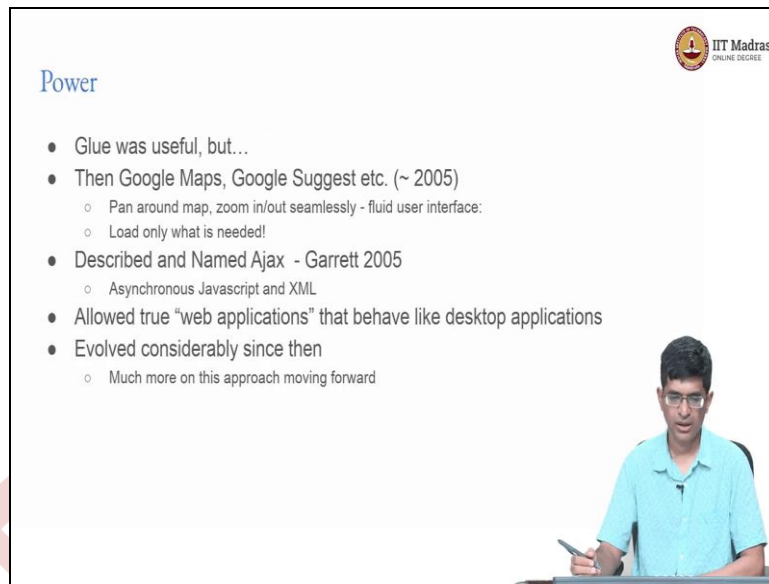
And the purpose of this language was primarily to be used as a glue language meaning that there would be different kinds of java applets or different kinds of other systems that are being assembled together in a web page and the primary purpose of the scripting language was to try and bind them together in some ways that you know enhance their functionality. So, it was not really designed too much for coding.

And one of the other things of course an implication of this is that because html is so forgiving I mean even if you have errors in your html it pretty much just goes through without really complaining. This scripting language also had to be sort of forgiving if the scripting language would keep bringing your web pages down and you know crashing saying oh this feature was not available.

People would get quite fed up of it and say look I do not want this language at all can I just have normal web pages and given the fact that it was primarily designed in order to assist the so called applets that were being designed in java it was called javascript which of course led to trademark complaints and people saying oh no you are trying to write off the popularity of java.

So, then there were a couple of other changes in name but the point is the name javascript pretty much stuck to the point where javascript today is as well known as java even though they have very little in common except for that part of the name. Now one of the main issues with javascript to start with was the fact that because it was created as this glue language and just sort of hastily put together to run in a browser it was slow and it has limited capabilities.

(Refer Slide Time: 04:45)



Power

- Glue was useful, but...
- Then Google Maps, Google Suggest etc. (~ 2005)
 - Pan around map, zoom in/out seamlessly - fluid user interface:
 - Load only what is needed!
- Described and Named Ajax - Garrett 2005
 - Asynchronous Javascript and XML
- Allowed true "web applications" that behave like desktop applications
- Evolved considerably since then
 - Much more on this approach moving forward

So, this part the glue language was useful but there was only. So, much that you could do but then somewhere around 2005 is when things really started changing quite dramatically it was partly a sort of an implication of a thought process rather than anything that changed in the language. It was not that the browsers changed or the language changed very dramatically.

But some people came up with some ideas on how to use javascript in order to enhance certain functionality and one of the best sort of demonstrations of this was in Google maps which came out around that time frame. Similarly Google also had something called suggest which would sort of type in you know give you suggestions as you type and the point about Google maps was you could pretty much of course most of you are probably familiar with Google maps at this point you can sort of you know scroll around in a map you can zoom in zoom out and it is all very fluid it just sort of shows you the part that you need to see.

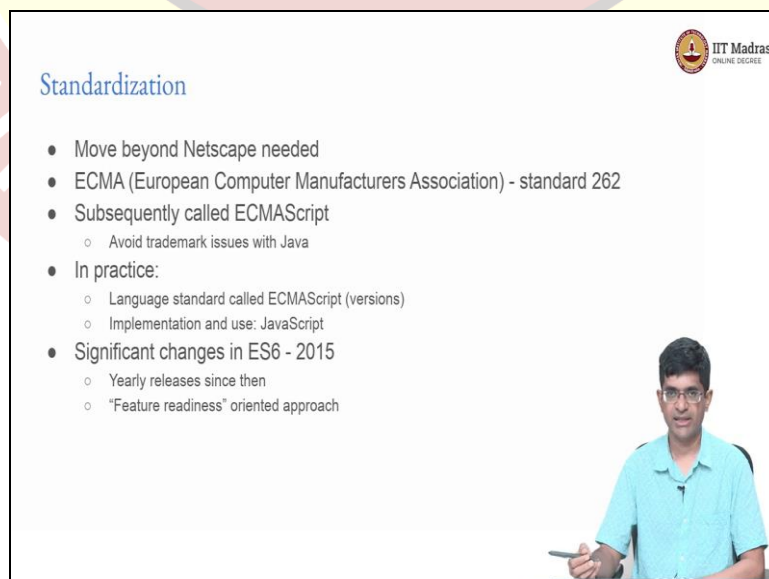
Now when you think about it how do you do that because you have a huge amount of data corresponding to the entire map of the earth at fine detail but you cannot load all of that information into a small device like a mobile phone or even a computer you can store the entire earth's maps conflicts. So, what was done over there was that as you move around on the screen the browser would keep sending back requests to the server asking for a new piece of data to fill in the blanks that are seen around it and this was all done.

So, seamlessly in the background using javascript that was where the power of javascript really came out. And this entire process was described by James Garret in 2005 and was called Ajax: Asynchronous Javascript and XML and the name also was popular the entire idea took off and this made a huge difference to how javascript was used in practice. In some sense it allowed two web applications that could behave a lot like desktop applications.

And the primary difference of course from a static application is that you would essentially have a much nicer user experience partly because the system would only load what was required in order to respond to a given interaction with the user rather than loading a new page each and every time. Now javascript has evolved considerably since then this was sort of the spark that started the fire but since then it has really caught on and spread.

And you know obviously at that point people started noticing deficiencies of the language and there were a number of attempts to fix certain problems. And we will be looking at much more of this approach this whole idea of doing things in the background and asynchronously loading material as we move forward. It is one of the main things that makes javascript powerful and suitable for the kind of web applications we are talking about.

(Refer Slide Time: 07:47)



Standardization

- Move beyond Netscape needed
- ECMA (European Computer Manufacturers Association) - standard 262
- Subsequently called ECMAScript
 - Avoid trademark issues with Java
- In practice:
 - Language standard called ECMAScript (versions)
 - Implementation and use: JavaScript
- Significant changes in ES6 - 2015
 - Yearly releases since then
 - "Feature readiness" oriented approach

So, now javascript had essentially reached a certain level of popularity which meant that it could not really be just tied down to Netscape one particular browser or even one

particular company. So, ECMA the European Computer Manufacturers Association developed what they call standard number 262 and eventually this came to be called EcmaScript ECMA script that was primarily sort of avoid trademark issues with java.

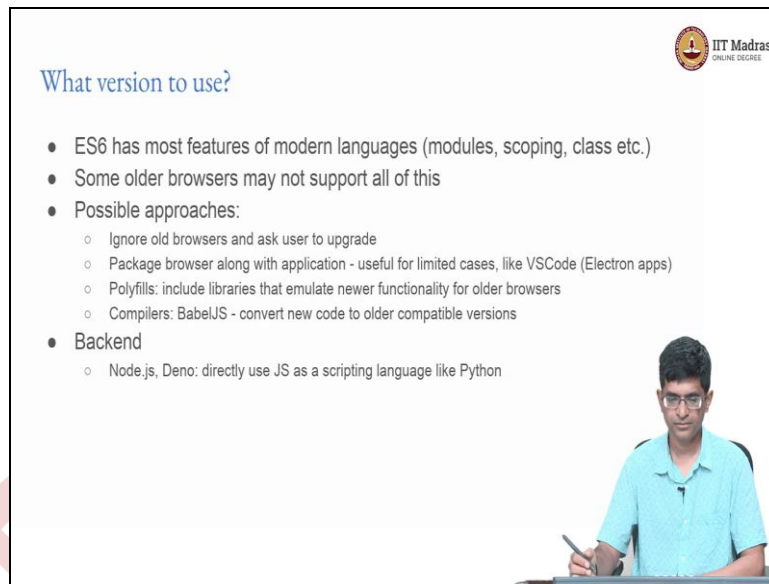
Nowadays in fact you may not even see the ECMA capitalized you may just see it as EcmaScript as an alternative name for javascript. Now in practice this language standard is what is called ecmaScript. So, we usually talk about ES versions right. So, there is ES4 ES6 ES5 ES6 essentially got established in 2015. This was a fairly major change that introduced a lot of new features in javascript.

But since then rather than saying ES7 or ES8 they pretty much just have a yearly release pattern where they say like roughly June or so, of every year there is a new release of certain features some things are standardized and frozen saying this is what we call ES 2020 or ES 2021. Now the point is because they have sort of fixed the time when those releases happen rather than talking about a specific release and saying you know does your browser support all of this in the release they took an approach which is called feature readiness where there is a technical committee that evaluates all of these features decides whether they are ready to be implemented.

And then you can say now this is part of the standard and the only question now is does a given browser actually implement a certain feature or not. So, given that ES6 has been around since 2015 for the most part in this course we will be looking at a reasonably advanced form of javascript. And we will be looking at in fact fairly new versions probably even things that have come up in years 2020 or 2021.

And part of the reason we can do that is because even if a browser does not support all of this there are compilers which do an interesting job of basically taking new features and converting them in such a way that they can still be run by older legacy browsers . So, in this course we will try to make maximum use of some of the good features that have been introduced into javascript. So, that we are not sort of stuck with workarounds.

(Refer Slide Time: 10:15)



IIT Madras
ONLINE DEGREE

What version to use?

- ES6 has most features of modern languages (modules, scoping, class etc.)
- Some older browsers may not support all of this
- Possible approaches:
 - Ignore old browsers and ask user to upgrade
 - Package browser along with application - useful for limited cases, like VSCode (Electron apps)
 - Polyfills: include libraries that emulate newer functionality for older browsers
 - Compilers: BabelJS - convert new code to older compatible versions
- Backend
 - Node.js, Deno: directly use JS as a scripting language like Python

And so, like I said which version to use becomes a big question. ES6 has the has most of the features of modern languages it has concepts like modules it has good variable scoping it has proper classes implemented at least the syntax for implementing classes is there. Some of the older browsers may not support all of this we have a few possible approaches we could just sort of ignore old browsers and ask everyone to upgrade their operating system and their browser.

Or in certain cases which you might have noticed in applications developed using the so called electron framework one example is the so called VSCode browser visual studio code from Microsoft. The VSCode is an excellent example of how an electron based app a web-based app can basically be made very powerful the entire thing if you look at it, it is running a chrome browser in the background but the chrome browser is not whatever you have installed on your system it comes packaged with VSCode.

So, for applications like that that are really going to be used that is a useful you know an acceptable solution but if everyone needs to package a browser along with their software it probably starts to make the system very heavy and you start running out of ram you run into various other issues out there. There **are there** is something called a polyfill which essentially is some way by which there are libraries that emulate newer functionality.

So, that it can still run on an older browser that does not have the same functionality and effectively what a polyfill is doing is related to this concept of what a compiler does


where basically it takes the newer constructs in the language and modifies them. So, that they can be run on older versions of the javascript engines in other words there are a number of different ways that people are found to work around the problems with versions.

And for the most part because of this we will be sticking to fairly modern features simply because that makes the language easier to understand than to use. Now the interesting thing is a browser is not the only place where you can run javascript somewhere along the line people realize that the language as such had grown to the point where it had some very nice and powerful features that even made it suitable for developing the back end which means that you could replace your python flask application completely with something written in javascript.

And running on usually what is called the node.js interpreter there is another interpreter called Dino which has some slightly more restrictive features but is also supposed to be more secure. So, node js is probably what you would find as the most popular backend mechanism for javascript at this point it's you know the main thing to note over there is it means that you are no longer limited only to a browser.


There are implications of this it means that when you are writing code in node.js there are certain things like the DOM may not be directly available to you are no longer always running in the context of a document. So, you need to keep that in mind when you are running node.js applications.

(Refer Slide Time: 13:29)



Implications of JS Origins

- Ease of use given priority over performance (to start with)
- Highly tolerant of errors - fail silently
 - Debugging difficult!
 - Strict mode: "use strict";
- Ambiguous syntax variants
 - Automatic semicolon insertion
 - Object literals vs Code blocks {}
 - Function: statement or expression? Impacts parsing
- Limited IO support: errors "logged" to "console"
- Closely integrates with presentation layer: DOM APIs
- Asynchronous processing and the Event Loop - very powerful!



So, like I said there are certain implications of the origins of javascript and the primary one was that ease of use was given priority over performance which meant that it may not always have been the fastest language to start with. This is interesting because today you know with all the optimizations that people have spent time on the engines that are present in modern browsers like chrome are actually very good which means that now you actually if I get fairly good performance out of javascript.

And not only that javascript is even capable of sort of interfacing with the graphic libraries and you know the graphic processing units that you might have available in your system. Now one of the biggest issues with javascript to start with was this notion that it could fail silently and why would people want to do that primarily because as I mentioned earlier html was forgiving.

Meaning that it would basically say that even if there's an error in your html source code you just sort of ignore it and move on try your best the browser should just try its best to display whatever it finds. Now if javascript also does the same thing it means that debugging becomes very difficult because you first of all do not know if something went wrong. If it did you do not really have much scope of finding out where it went wrong because the browser is sort of just tolerating everything that it can.

Nowadays of course given the extent of use of javascript this has changed and there are certain things for example like the use of the use strict mode which is not just possible but strongly encouraged when you are writing scripts today which will allow most of

these constructs of javascript to at least get flagged at the development phase. It might still go through and finally run in sort of best effort basis on a browser.

But it would generate a lot more debugging errors that as a developer you would be able to look at and understand with more clarity. Now the other problem with the javascript origins is because it was sort of a quickly put together language there were a number of places where the language had sort of ambiguous syntax. There is this one concept called automatic semicolon insertion which basically says that you do not really need to terminate each line with a semicolon.

The problem is that is a bit vague it says you do not need to but there are a couple of instances where you can actually run into problems if you do not properly put in a semicolon. So, unlike a language like C which is very clear that you have to put in a semicolon after every statement or a language like python which says that no semicolons are irrelevant they are not even part of the language if you put them at all they will just sort of terminate a statement.

But you should you cannot sort of separate statements just using the semicolon you need the indentation. Javascript sort of says yeah you know you can put semicolons when you want to but if you do not then the interpreter will figure out where to put them not the best of approaches and I would strongly recommend that you follow a consistent approach and in the case of javascript the probably the best solution is put a semicolon after every statement that is that gives clarity.

At least there is no cause there is no chance that there is ambiguity in the code that you typed in. There are also a few other problems for example object literals and code blocks both use this double bracket notation the bracket notation and there is also this thing about what exactly is a function. When you declare a function is the declaration a statement or is it an expression with the return value with the return value. Why are all these important because they can impact how you parse a given javascript file.

So, these ambiguous syntax variants of course they are coming down and they are sort of being deprecated and discouraged as we move forward. So, when you are learning javascript fresh please try and avoid any sources of ambiguity be aware of what these

possibilities are. There are a number of very useful tools usually called linters and a linter is something which basically goes through your source code and gives you suggestions on how to improve the readability and maintainability of the code.


It is a very good idea to get used to using some of those. Now another implication of the origins of javascript is that it has very limited support for input output because after all you are running inside a browser you do not really have a way by which you can sort of communicate with the user directly the javascript language cannot. Yes you can have forms you can do something else of that sort but the language by itself does not have a straightforward way to communicate with the user which means that all errors have to get logged into something else called the console.

And the developer then needs to look at that and that console may not be visible directly to an end user. Now having said all of this these are sort of the drawbacks of javascript there are a couple of other things which make it very powerful one of them is the close integration with the presentation layer the so called DOM. The document object model and the api's for interacting with the DOM.

Mean that javascript has a way by which it can directly manipulate what is seen on a web page and this is immensely powerful this is what finally leads to the actual use of javascript for developing interactive applications. And apart from that there is this concept of asynchronous processing and the event loop that we will be looking at in a little bit more detail later.

Asynchronous processing is one of the things that really makes javascript powerful and ultimately also led to its use in the back end because it allows some very neat sort of ways of expressing programs that can be used in different ways.


(Refer Slide Time: 19:21)



Using JS

- Not originally meant for direct scripting
 - Usually not run from command line like Python for instance
- Need HTML file to load the JS as a script
 - Requires browser to serve the files
 - Links and script tags etc.
 - May not directly work when loaded as a file
- NodeJS allows execution from command line

Examples here will be shown on Replit for the most part



So, now comes a question how do we use javascript and since I mentioned that you know it was not meant for direct scripting. So, it is not like you take a text editor you write a file and like for example with python you would just say python and the file name and it would run to start with a case there was no interpreter for javascript that you could call in that way. Nowadays there is there is no JS but we will get to that later at some point we are not really going to spend much time with node.js we are interested in the front end.

Now in the context of the front end what happens is that you need an html file that is actually going to load the javascript file as a script and execute it. What that means is you actually need some kind of a browser well actually a server to serve the files and the browser that can actually open the files and make the requests to get the files. You also need to embed the javascript itself using some kind of the link notation that is there in html.

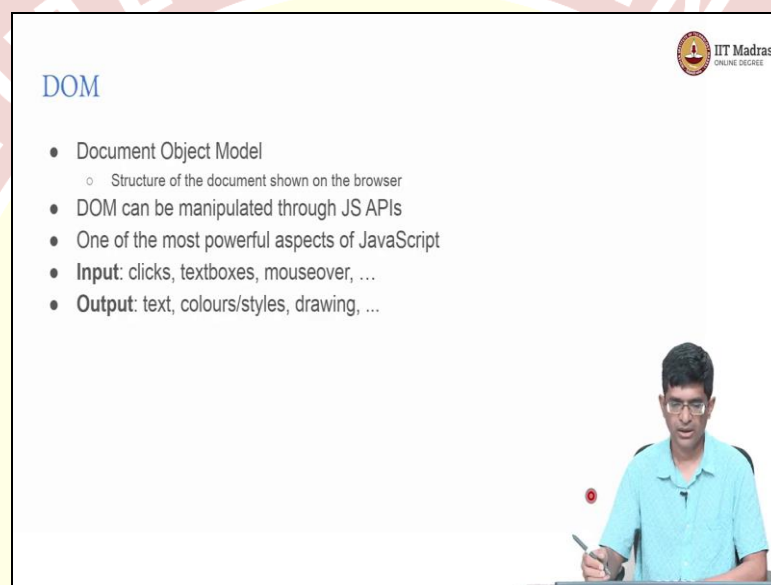
Now the implication of this is depending on the browser it may not work if you directly load the file from your file system. It may or it may not but it also depends it finally comes down to whether or not it has the ability to access the javascript file also directly by loading it like a file. If it needs to actually make a separate request for that it is probably not going to do it.

Now what this means is that for you who's as a person who's trying to learn javascript you need to sort of figure out what is a good platform that will allow you to do all of

this. And in our case at least for the demonstrations that I will be using throughout this course or at least in the beginning parts will be using replit which hopefully most of you have some familiarity with.

And what we will do is replit has this nice sort of starter code that allows you to basically create the starting html put in a CSS file and also a script file in javascript where you can put in anything that you want . So, we will essentially be using that as our starting point in order to go through and understand the basic use of javascript.

(Refer Slide Time: 21:29)

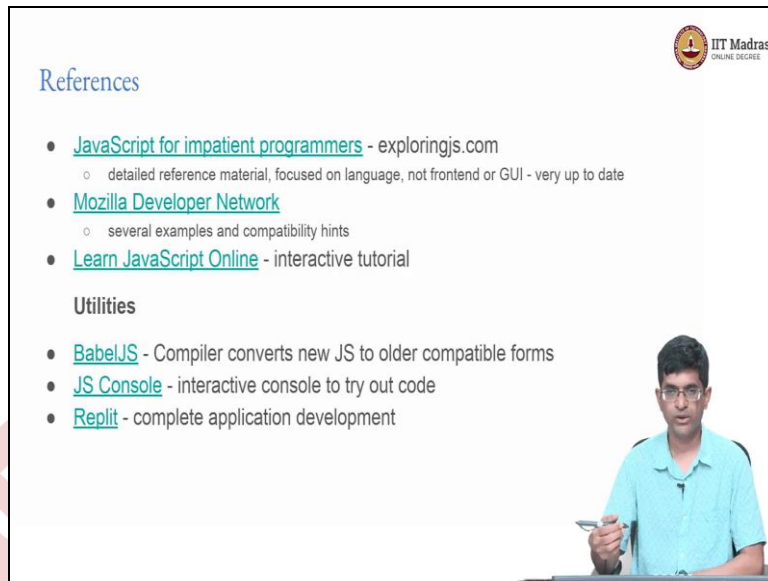


Now like I said the document object model is one of the things that makes javascript useful it is basically the structure of the document as shown on the browser and it can be manipulated through JS api's. Not only can you change the text that is inserted or displayed at different parts you can even change the structure of the document itself and this is pretty much one of the most powerful aspects of javascript.

Now the DOM also has some ways by which we can interact and there we basically have some kind of input where you can click or you can have a text box where you enter data even just moving the mouse around can be used as interacting with the application and it also has output where basically it can display text it can change the color of the style of text or it can even draw things on the screen.

HTML5 has this tag called canvas which basically allows you to draw arbitrary things onto the screen.

(Refer Slide Time: 22:24)



References

- [JavaScript for impatient programmers](#) - exploringjs.com
 - detailed reference material, focused on language, not frontend or GUI - very up to date
- [Mozilla Developer Network](#)
 - several examples and compatibility hints
- [Learn JavaScript Online](#) - interactive tutorial

Utilities

- [BabelJS](#) - Compiler converts new JS to older compatible forms
- [JS Console](#) - interactive console to try out code
- [Replit](#) - complete application development

So, now with all of this I am going to end this portion over here with just some references there is this website exploring js.com where there is one of the books that is available over there is what is called javascript for impatient programmers. Now this is a very good reference it has very detailed reference material it is focused on the language which means that you do not really get too many examples of how you can use it for manipulating things on the front end or you know the interacting with the DOM is not really covered in this book.

But it is a very good reference for the language itself. Now the Mozilla developer network this is something that we have seen even app dev one. It has several examples compatibility hints and so on you know things that sort of give you a lot of different hints on how what kind of aspects of javascript programs can be used and also examples of how to use them.

There is a very good interactive tutorial called learn javascript online the first several chapters of this are free but the more advanced topics you need to pay for I mean there are it is on a page basis. And apart from this of course you know you pretty much search online you are likely to come up with a large number of tutorials. So, you know pay attention to the tutorials please be aware that not all of them are of the same high quality.

Some of them might sort of even encourage certain bad practices in terms of code primarily driven by the fact that you know that is how code used to be written until

certain good features came along. So, you will need to sort of apply your mind and say you know is this really the best way to write code or is this a good reference. These are some that I can think of that are good starting points.

But at the end of the day it comes down to you as the user you might find some other document or some other tutorial to be more useful for you. Apart from this there are a few utilities that are worth pointing out I had also already mentioned Babel JS it is a compiler that converts the newer versions of javascript newer forms of javascript things that might not be supported in some of the older browsers into forms that are compatible with older browsers.

So, it is basically a compiler or in some cases people might even call it you know a transpiler a translating compiler. Basically what it does is it takes javascript as input and produces javascript as output and the idea behind doing this is that the output javascript is guaranteed to be something that will run on even an older version of a browser it does not have like requirements that your browser supports the latest ES2021 features and so on.

It can be useful from the point of view of learning maybe not so much from the point of view of implementation very likely, yes. There is this website called jsconsole.com which basically provides you with an interactive console where you can try out code but to be frank you know every one of you actually has direct access to a console directly in the browser. So, anything like chrome for example it has the so called developer tools.

Once you activate that it basically brings up a small console where you can type in basic javascript statements and see their impact like and because chrome is a browser that sort of you know keeps fairly up-to-date with the latest versions of javascript and so on. It is actually you know a good way of testing out most of the features that are going to be supported in any reasonable modern browser.

And of course; finally replete which I will be using for some of the demos and has good support for basically working with javascript applications.