INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

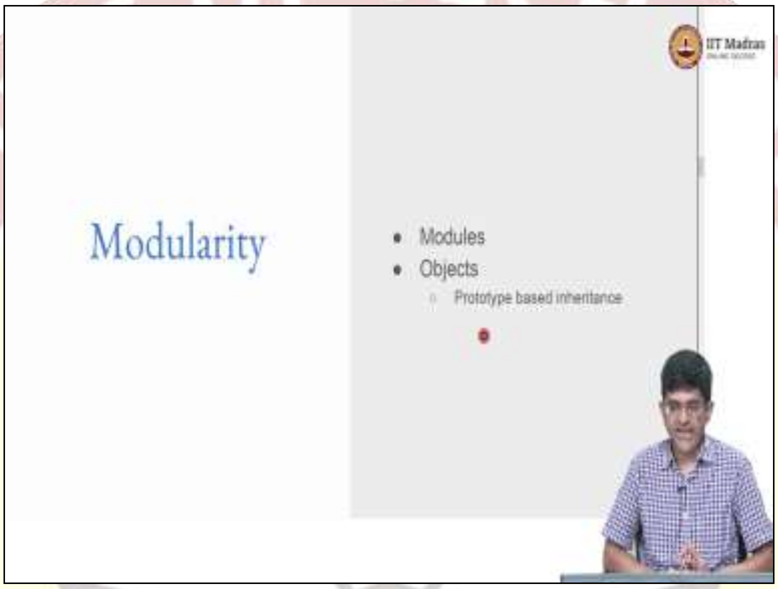
IIT Madras

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 - Modularity

(Refer Slide Time: 00:14)

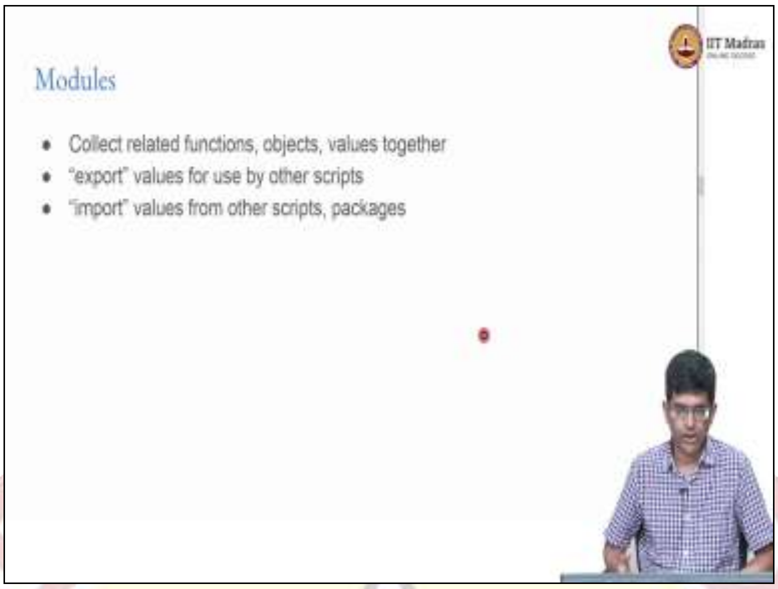


Modularity

- Modules
- Objects
  - Prototype based inheritance

Hello everyone welcome to modern application development part two. All right now we have seen objects in the form of arrays not really objects in a generic class type. Right so, far what we are going to do is now spend a little bit of time trying to understand Javascript's notion of objects. But before that we also want to understand a little bit about how Javascript allows you to encapsulate things into modules right and modules are very important from the point of view of reusability of code okay.

(Refer Slide Time: 00:41)


Modules

- Collect related functions, objects, values together
- "export" values for use by other scripts
- "import" values from other scripts, packages

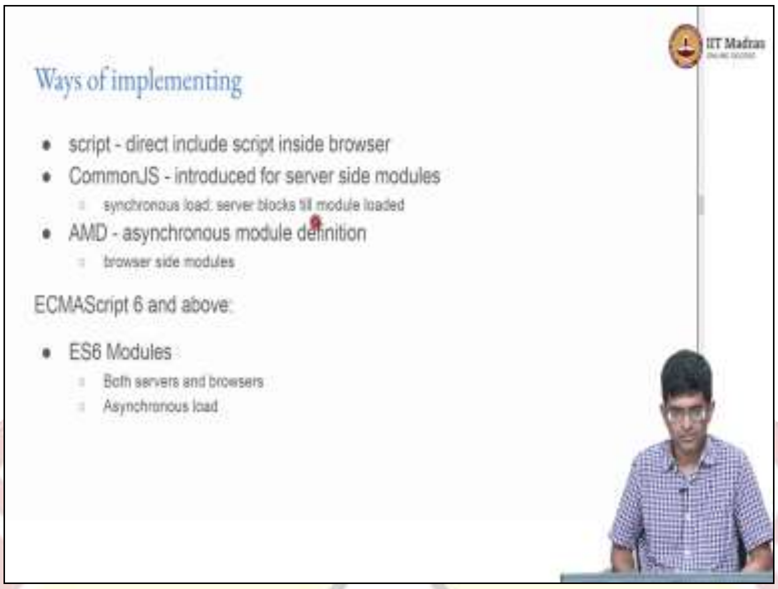
The whole idea of a module is that you collect related functions or objects or values together from python you might have come across the numpy module for example which basically collects a lot of numeric python related functions together right scipy is for scientific python okay. There are others for csv for example right for dealing with comma separated values. And the idea is that you have a whole lot of different functions and classes that are all related and you bundle them together into one or more files that can then be imported by others and used okay.

In Javascript same idea right you explicitly export certain values by for use by other scripts and you can then import values from those other scripts into your code okay. Now Javascript has one problem which is the fact that a lot of Javascript is used on the front end which means that you need to be very clear about how you are going to import are you supposed to be importing from a local file where is the package present is it present somewhere on the net.

Should the package be downloaded if required right and how do you sort of generally do this relative addressing of packages and so on okay. On the other side Javascript is also used on the back end using the node.js interpreter and when that happens well you know you can use it just like python right you have the packages you define directory hierarchies you put files in certain directories and you can import them as required.

So, because of these two different ways of using it we have to be a little careful about how we package things and how we are actually using the code in our in our programs.

(Refer Slide Time: 02:26)



Ways of implementing

- script - direct include script inside browser
- CommonJS - introduced for server side modules
  - synchronous load: server blocks till module loaded
- AMD - asynchronous module definition
  - browser side modules

ECMAScript 6 and above:

- ES6 Modules
  - Both servers and browsers
  - Asynchronous load

Now what happened is when Javascript started out there was no notion of modules right and it basically just had scripts. You directly include scripts inside a browser and those scripts execute okay. Now once people started using server side Javascript there was this notion of common js right which was brought in, in order to implement what is called server-side modules they wanted to actually package it and start using it more like an interpreted language.

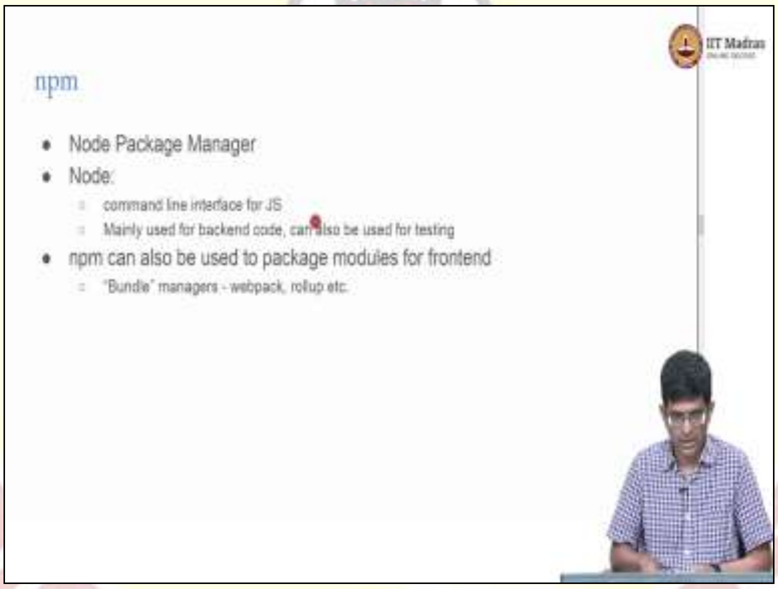
Now common js has some issues one of the main ones being that it uses synchronous loading right. And synchronous loading means that it basically the moment it comes to the import statement it waits until the import has completed before proceeding right which is fine at the server end because after all you know it is running once it imports and then it runs from there. But the moment you do anything synchronous or blocking as it’s called on the browser it’s a big problem because it actually makes the user experience very bad okay.

So, any kind of synchronous loading or blocking loans on a browser are basically unacceptable right because there might even be a possibility that you are trying to do a load from some server that is down and what will happen to the end user that page will just be completely frozen because Javascript works that way we will get to that later. So, on the client side on the browser side there was another definition called asynchronous module definitions or AMD, AMD modules now ECMAScript 6 which was sort of formalized in 2015 and has been developing since then brought in this notion of ES6 modules works on both servers and browsers it uses asynchronous load.

There are certain packages and possibly even certain browsers that do not fully support this although browsers should pretty much support it at this point what you might find is there are some relatively old packages that do not support ES6 modules completely as they should. Now as far as we are concerned we are going to go with the latest version of the language because it's not really worth sort of you know going back and learning old version of the language just to sort of keep up with those things.

If you do encounter an older piece of code that you need then you might have to sort of understand a bit more about how modules are written there in order to make use of it.

(Refer Slide Time: 04:43)



npm

- Node Package Manager
- Node:
  - command line interface for JS
  - Mainly used for backend code, can also be used for testing
- npm can also be used to package modules for frontend
  - "Bundle" managers - webpack, rollup etc.

Now just a word about npm which is the node package manager right, node is a command line interface for Javascript it is mainly used for back-end code it can also be used for testing right. And the interesting thing is npm can also be used in order to package modules for use at the front end right and this is usually done using. So, called bundle managers right there is something called web pack there is another one called roll up there are a few different things that allow you to take the packages that were defined using node modules.

And put them into different js files which can then be directly included or imported even on the client side okay. Now for the most part we are going to ignore all of this we don't need to know about it now but at some point as and when you start actually you know bundling the thing together into an application you need to know a little bit more about how these work okay for now we are ignoring these aspects okay.

(Video Start: 05:45)

So, let us first look at modules right in Javascript we look at some examples going back here. So, now let me look at the index.html first and what I have is right if you look at this you will find that there is this line 16 which basically says script source equal to modularity.js and this is basically the file that I am going to show you. And it says that it’s a type module right.

And it turns out this is important because if I do not have this type equal to module here I cannot use import statements or exports for that matter inside it okay. Now more importantly if I go look at modularity.js I will find that I am only going to be importing things from here right and you know I then have the regular Javascript functionality because I have this import statement out here in the index.html I needed to have this type equal to module okay.

But where am I importing from if I look at modularity.js I will find that I am importing from module1.js right which contains this code. So, let me say that you know I export this right but if I go look at the index.html module1.js is nowhere in the picture I don’t have dot to put it as a script right modularity.js automatically picks it up from the place that it is needed and runs it.

So, what happened in module1.js I just have a single line export const c equal to some number right and in modularity.js I import c from module 1 and print it out what happens when I run it as expected it you know got the value and printed it out okay. So, this is the most basic form of how I can import something from another module right but why would I do this I could have just declared const c in you know the other file to start with instead I can do something else which is that let’s say I do this right which is oops.

This top language I define one function right. So, in module1.js in other words I have a function which is s q of x which will square a value and I also declare a const c which is exported. So, this can be used somewhere else and another function energy which can also be exported and which makes use of this sq of x function. So, both of these the value of c a const and this function energy are being exported.

And I am going to import both of them out here import c, energy what will happen when I run this console.log of c sure enough I get the value out here what happens if I console.log energy I get that value as well over here. Now of course I probably need to try importing the value sq as well before I can actually run console.log sq of 10. But what happens when I run this it says the module does not provide an export named sq.

So, because I did not have export in front of this function sq it does not allow me to import it over here. So, I have to basically lose this if I try running it, it is just going to say sq does not exist right and when I finally run only these two yeah sure enough that works. So, this is sort of the most basic form of importing. I can do something further which is basically I can rename an import which is to say that when I am importing c I can rename it as speed of light.

And from there onwards I have to use the value speed of light in my code. So that is just in case you want to sort of change the name for internal use. And now there is something further that I can do over here there is also another format in which I could do the same thing which is to say I will that this as well. So, now what I have is if you look at this way the code is written I have const speed of light equal to this function e of m equal to this.

But now when I am exporting I am saying export speed of light as c and export e as energy and what I am saying is I am sort of renaming the export. So, that when I finally go back over here and I run this same thing I import c, energy and I run it, it just works exactly the same way as before. Because as far as module1.js is concerned even though internally it thought the value was called speed of light.

When it exports it, it calls it c. So, as far as the other file is concerned it imports it as c. Now there is also a notion of something called a default export and what we can see over here is in this module1.js I have declared a value I am not exporting it and I am exporting a default function I am not giving the function a name normally it would have been function energy of m instead of that I am just saying export default function.

In other words this particular module is exporting only a single thing in this case it happens to be a function. How would I do this I cannot then do import c, energy because they are not being exported but if I just do import energy right it takes whatever was the default export from module1.js and gives it the name energy for use inside my code and I can directly run it and it would give me the value that I expect.

And there is one last part which is actually quite interesting to observe and this worth understanding in terms of how modules actually work. I am exporting two things over here one is a value x and the other is a function ink x. Now this value has been declared using let not const ok which means that I should be able to change that value and sure enough this function ink x is actually able to modify that value.

Now let me go to this modularity and say I am going to first import x and ink x sounds good log the value of x what should I get I should get the default value which is 0. Now after all x was imported x was declared using let can I change its value? No it says assignment to constant variable. In other words even though x was declared as the let inside the module when it comes inside when I import it I get only a read only view of exported variables.

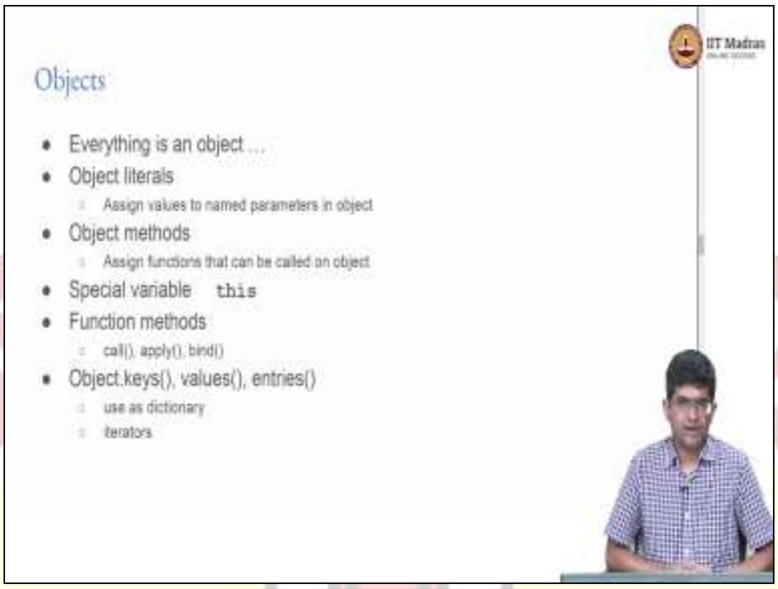
Does that mean I can never change it again no I can call the function in case and by calling the function incx() what happens well I should have printed out the value it does change the value to 1. So, in other words the only way of changing a value that is being imported is by calling the functions. So, that that were also written in the same scope as the value that was being exported and by doing that we can basically have this kind of you know the behavior that we want.

Where we are actually have a value that changes over time and can be modified but cannot directly be modified using something like an x plus plus over here. So, that is an interesting thing to keep in mind when you are sort of exporting variables. Now in general of course you need to think a lot before you create a module or exports or anything of that sort. So, please don’t just you know take this and say oh yeah you know maybe I can export it like this and not use it the other way.

Before using any of these exports or imports you need to think clearly about how you are structuring your code to start with.

(Video End: 14:29)

(Refer Slide Time: 14:30)



Objects

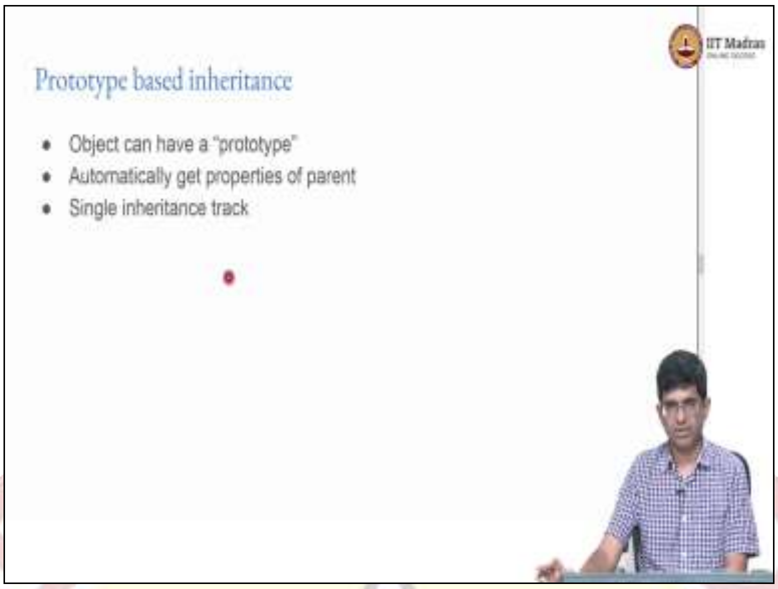
- Everything is an object ...
- Object literals
  - Assign values to named parameters in object
- Object methods
  - Assign functions that can be called on object
- Special variable `this`
- Function methods
  - call(), apply(), bind()
- Object.keys(), values(), entries()
  - use as dictionary
  - iterators

So, we looked a little bit at what modules are and how they are implemented. Now let us understand a little bit more about objects in Javascript and over here the thing is everything in Javascript is an object. There is this notion of object literals that are used to assign values to named parameters in object in an object and there are object methods which are functions that can be called on the object.

There is also the special variable which is labeled this which is found inside objects we will get to that in a moment. There are also certain other functions like call apply bind and so on which are again interesting to know they are sort of functions of functions. So, to say right and as mentioned earlier there are certain standard methods that belong to object the parent class like keys values entries and so on.

So, you can use an object as a dictionary and in this case you know these things allow you to sort of iterate through the values in the dictionary.

(Refer Slide Time: 15:33)



Prototype based inheritance

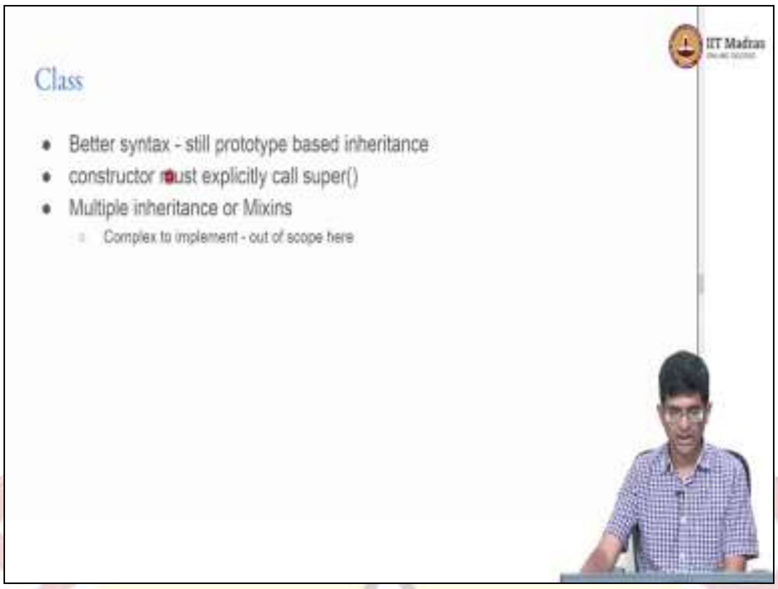
- Object can have a "prototype"
- Automatically get properties of parent
- Single inheritance track

Now Javascript has a notion of object orientation but not exactly in the same way that other languages like java for example or even python have. So, this notion of class in Javascript first of all the keyword class now exists and you should use it because it sort of makes the coding much cleaner. The point is that it actually works using some something called a prototype.

So, rather than saying that an object is inheriting from another object what you I are actually saying is I can just create two objects and I can set the prototype of one object as the other object which means that it automatically sort of gets all the behavior of that parent object. Now this also means that there is sort of a single inheritance track you always have one direct parent relationship unlike the sort of multiple inheritance that are there in other languages.

Now in most cases when you are using things like multiple inheritance it is something to be done with a lot of caution in any case. So, the fact that Javascript does not allow this is not necessarily a sort of showstopper problem. It does mean that you need to change the way you think about certain things and in a lot of cases you will find that you know you can actually get by without necessarily having to use things like multiple inheritance. And this prototype based inheritance is actually quite powerful in itself.

(Refer Slide Time: 16:55)


Class

- Better syntax - still prototype based inheritance
- constructor must explicitly call super()
- Multiple inheritance or Mixins
  - Complex to implement - out of scope here

Like I said there is this notion of a class which primarily acts as a better syntax and a cleaner way of writing the code you have this notion of constructors and in this case they must explicitly call the parent constructor. There is a way of implementing mixins or multiple inheritance but it is somewhat complicated to implement and definitely we are not getting into those details over here.

(Video Start: 17:22)

So, let us look at some examples of implementing objects in Javascript and what we will do is start by you know just creating a most basic object it just prints out you know what the object is. Now the interesting thing is I can do more I can directly add a function as one of the elements in the object I can basically say xx.add is equal to this. And what happens when I print this out I can say xx is of type object xx.add is of type function.

And when I evaluate xx.add with 3, 4 it actually gives me xx dot 3, 4 is equal to 7 that is in other words it calls this function adds 3 and 4 and gives me the output. So, in other words xx dot add is now a method of this particular object we do not have classes yet it is just an object with its own methods. Now I could go further and say I declare a new function xx dot f is equal to just an anonymous function a function of x which returns this dot a.

So, what is this dot a the parameter this over here refers to the object and of course I mean it is like the self in python or this itself in java. It refers to the object and this dot a as we know over here has the value 5. So, when I run f with a parameter passed into it I should get 5 added to it as the result and yes sure enough we can see that 10 + 5 is equal to 15. So, in other words this over here is a useful parameter to have which allows us to refer to the object itself and to other parameters of the object.

If I directly tried saying return a plus x it would not work because it does not know what a is there is no variable called a in this present scope. Let’s just count this out now one a few interesting things especially you know they can be tricky. So, you need to keep this in mind what happens when you copy an object? I say x is equal to a 1 2 y is equal to x and obviously when I print them out I get the same value so, far so good.

But now I assign a new value to x dot a and I am going to once again print out the values of both x and y and what do I see both x and y have changed the a has become 3 in both cases. In other words y over here did not actually get the values of x it is pretty much just a pointer to x or a reference whatever you want to call it, it is the equivalent of just basically taking a pointer.

So, what does that mean does that mean that we cannot take copies well here comes the spreading operator again right. When I say z is equal to dot dot x it spreads the values of x out. Now I change the value of x dot a print x print y and print z and what do I see? I find that finally let me get rid of the other console logs because we already know what those are. The last three values alone x now has a equal to 5 y still has a equal to 5 because after all y is just a pointer to x.

But z does not z has the value 3 because it was done as an explicit spreading of x which means that it made sort of individual copies of those values . So, when you want to copy an object in other words you need to be careful. How you are doing it this is not the end of the story there is some notion of shallow copying and deep copying which is that if there are further objects inside the object itself it is not going to sort of go deep and copy each one of them individually.

So, that is something that you do need to keep in mind whenever you want to make a copy there are certain helper functions that allow you to that help to basically make complete copies of objects when you are working with them you need to make sure you are using something of that sort in other words just saying the assignment y equal to x is not good enough when you are trying to use objects. It only gives you a pointer and not a complete copy of the object.

(Video End: 22:04)