From 1 to 19

* document.getElementById:

document.getElementById(“ID”)

* document. getElementsByTagName(“Tag\_Name”):

document.getElementsByTagName(“p”) -> return list which can be access using index

document.getElementsByTagName(“\*”) -> returns all tags in document

document.body.getElementsbyTagName(“\* or etc”)

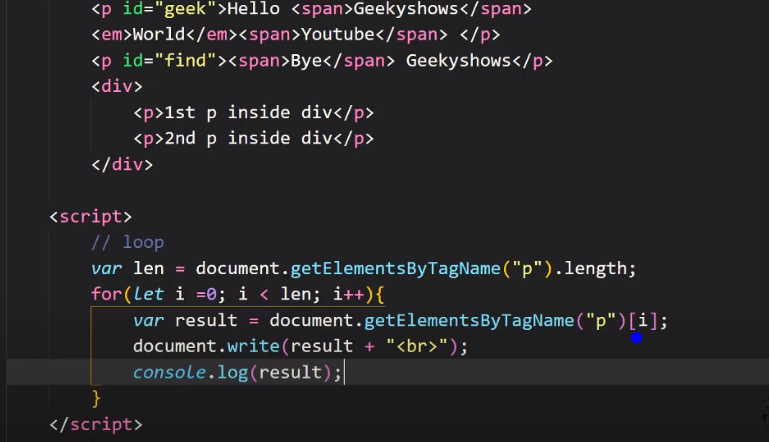
More Specific:



Length attribute:



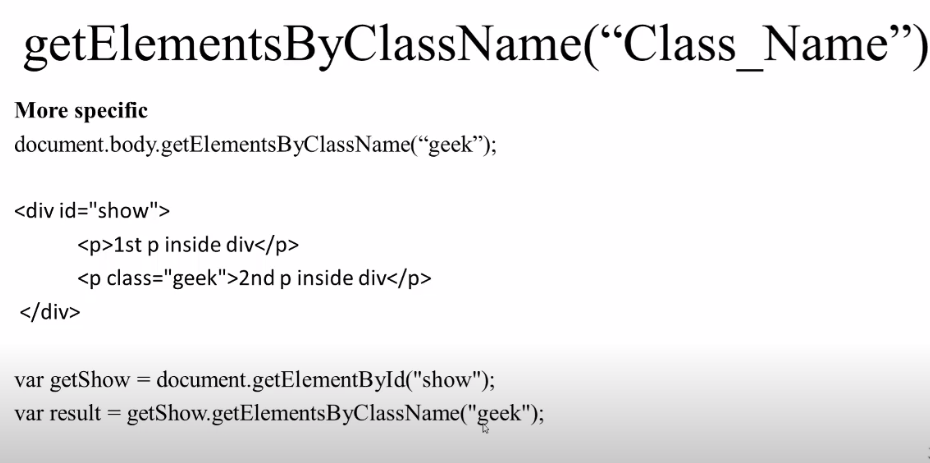
For loop:



* document. getElementsByClassName(“ClassName”):

document. getElementsByClassName(“ClassName”) -> again returns list access by index

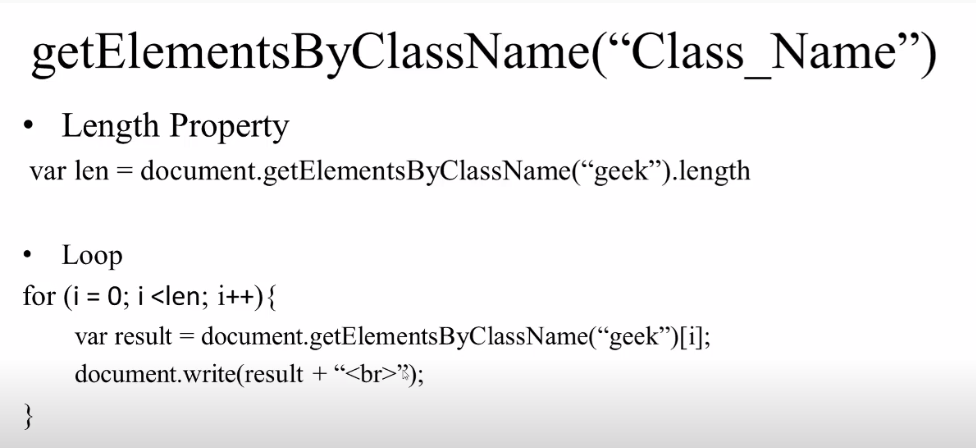
More Specific



For Length:

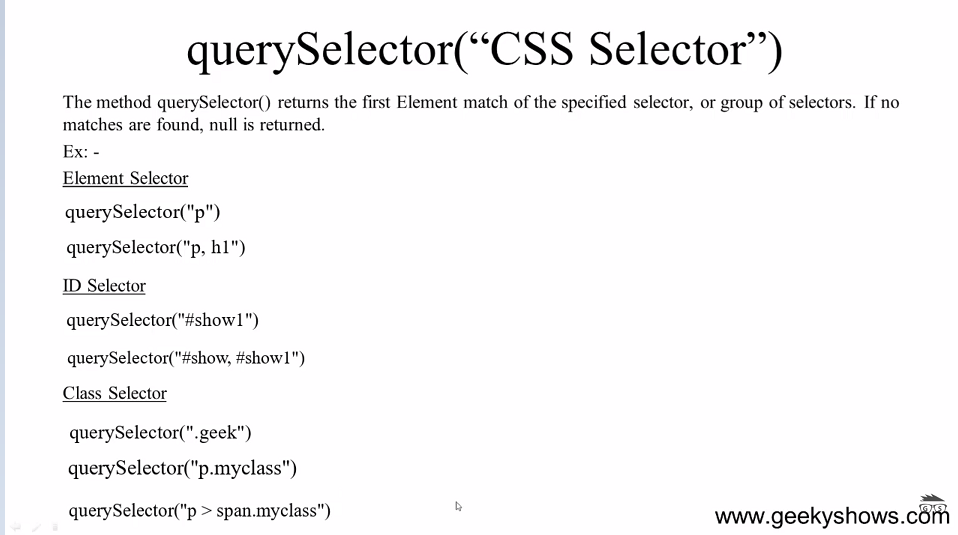


For Loop:



* querySelector(“single return”):

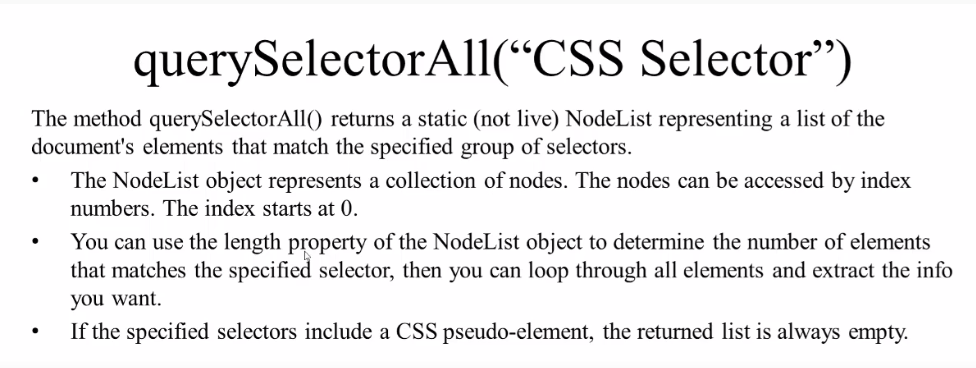
Returns only first matched value, not all. First come first get, whether we pass multiple values separated by comma or single, it returns only first value it sees In the code. For all, we use querySelectorAll

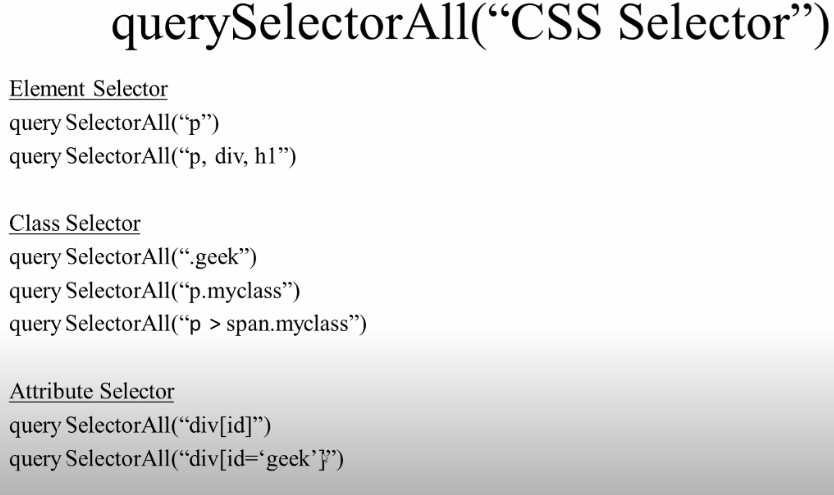


Multiple selection should be separated by comma, like if querySelector(“p, h1”), it returns first element in code only, like if h1 earlier before p, then it returns only h1 and ignore p

querySelector(“p.myclass”) -> returns tag of myclass in p tag

* querySelectorAll(“multiple returns”):

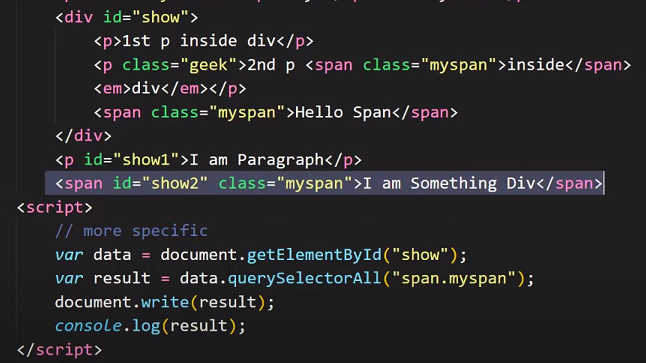




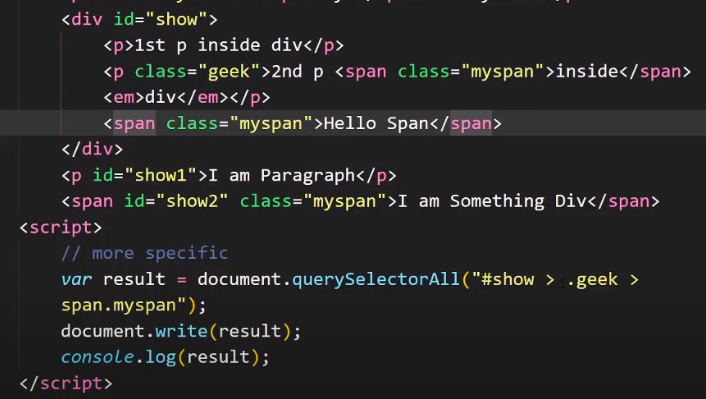
querySelectorAll(“div[id[“) -> returns all that div which has attribute id

More Specific:

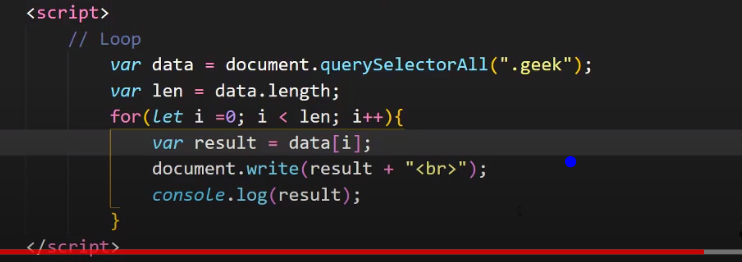
Find id show and then span tag with class myspan



Another way:



For Loop:



* WebPage Properties:

In console of any website, write

Console.dir(document) -> it will return all properties(like url, domain title etc) use in that website

Then from that, we can access each property separately in VS or console as;

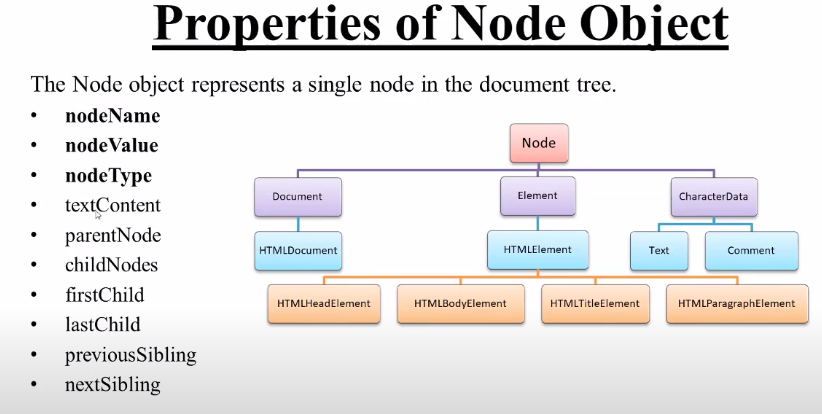
document.title or document.domain

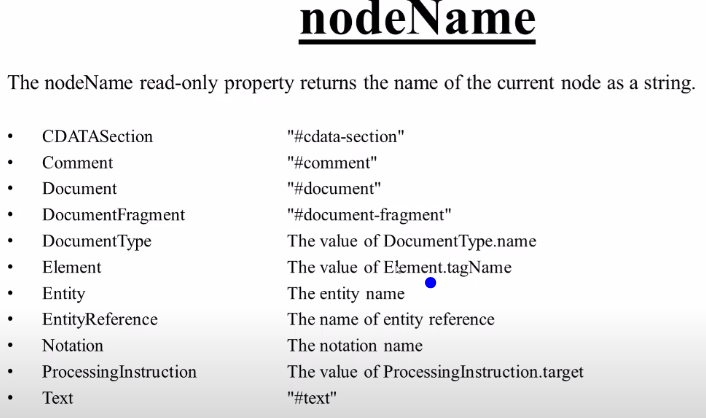
(In VS -> under script write, console.log(document.title) )

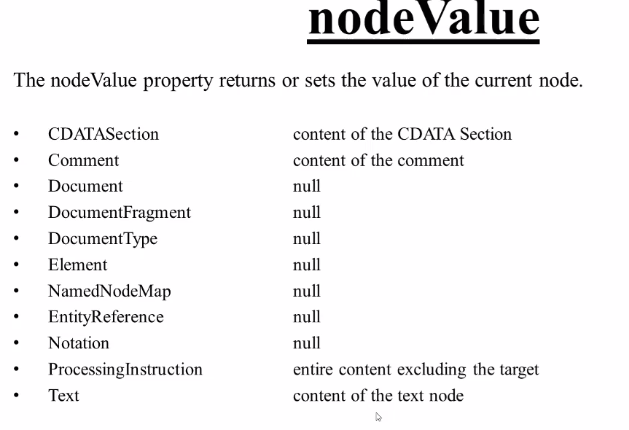
And we can also update those properties in js, like we can change title of website as

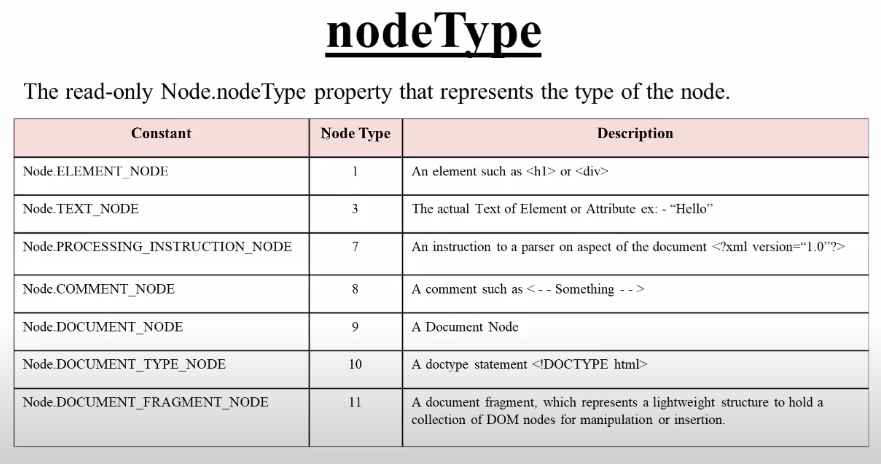
document.title = “Update title”

* Node Object:





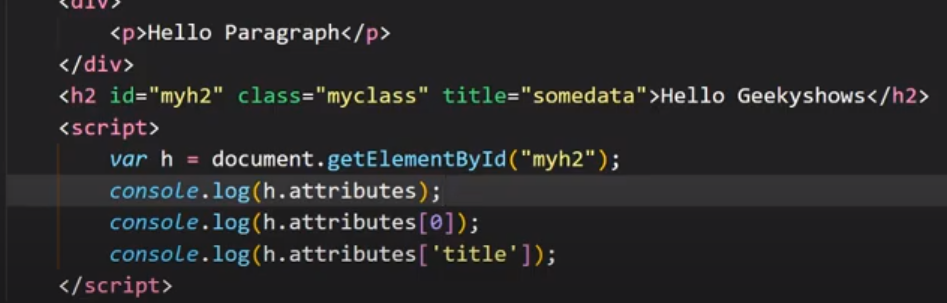




Attributes property in JS:

Name.attributes -> returns all attributes in a tag

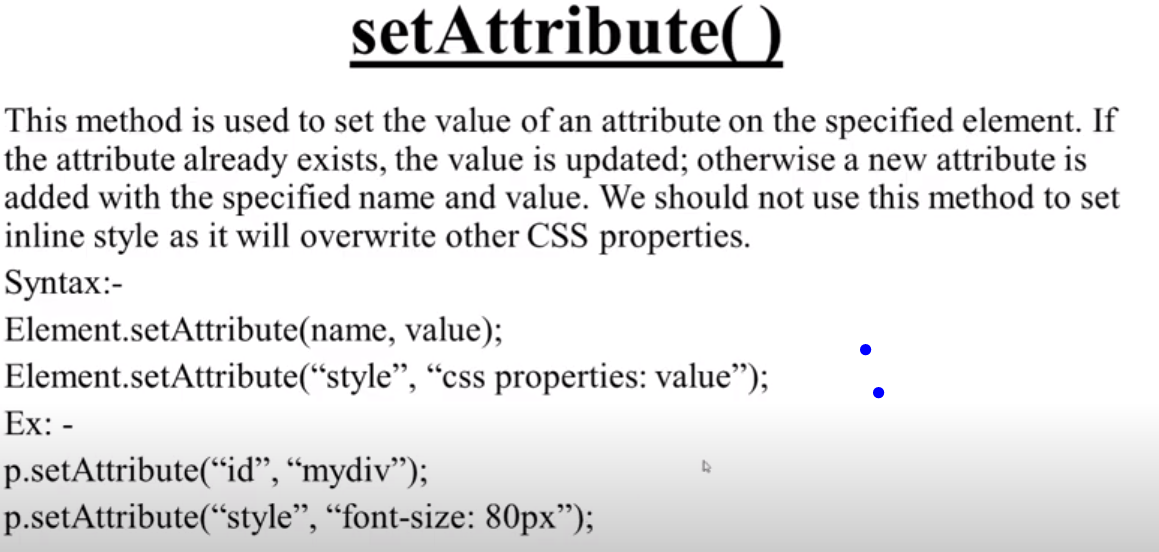
We can then access them using list indexing or key value pair



From 60 to 70

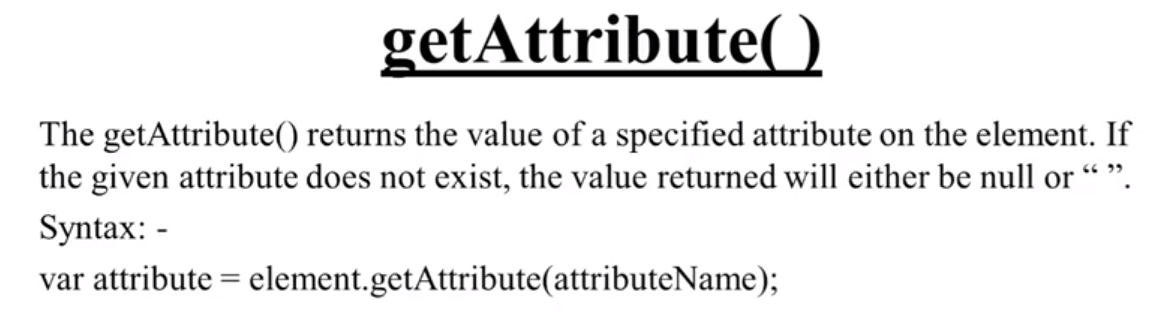
* Set Attribute:

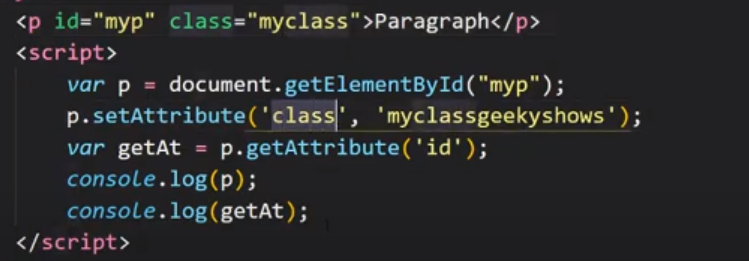
It updates value of given attribute of an element if it is already exist or create new attribute with the provided value if not exist. Remember It will overwrite the previous values of the provided attribute



p.setAttribute(“id”,”mydiv”) -> set id=”mydiv” of p element or p tag

* get attribute





Other method of setting attribute:

Like if we want to set a class then;

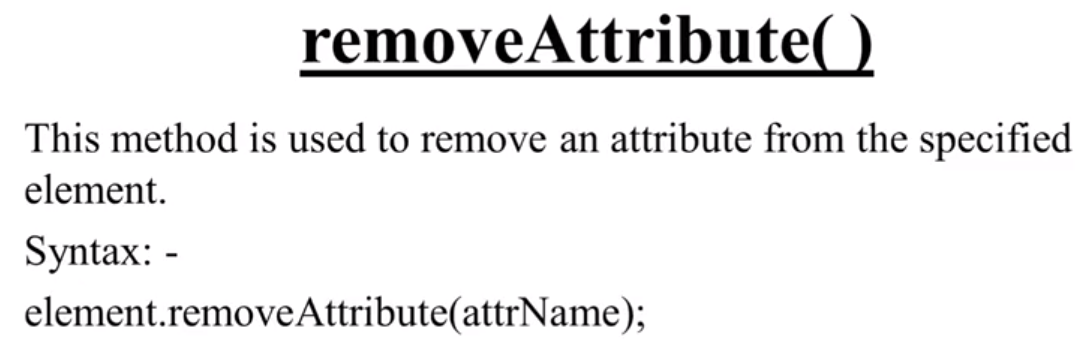
p.className = “myclassgeekyshows”

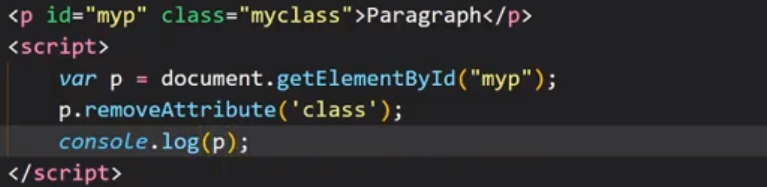
document.getElementbyId(“myp”).className or .title etc

and similarly getting an id can be;

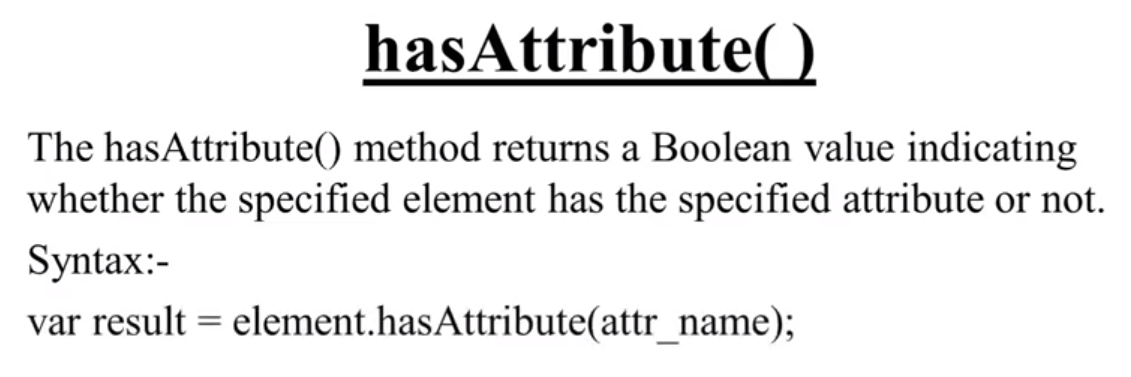
console.log(p.id)

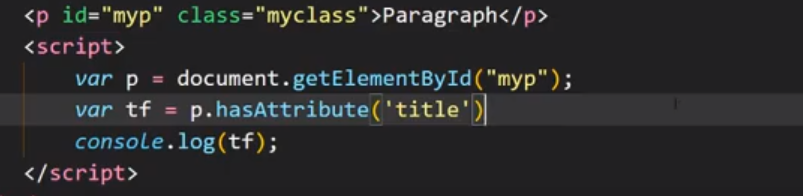
* remove Attibute:



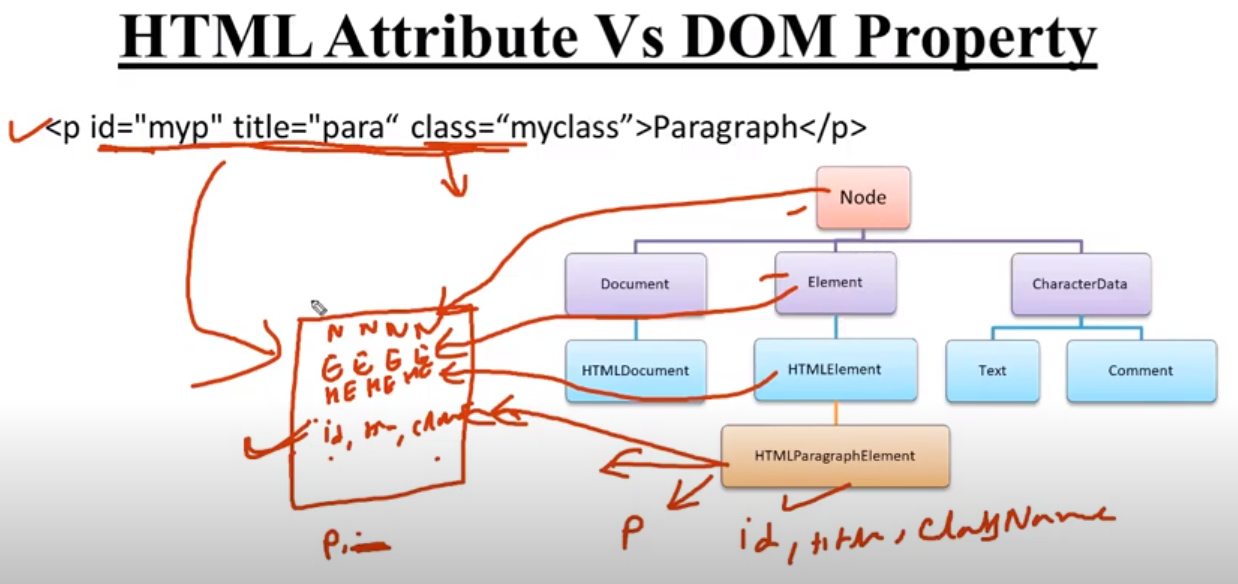


* Has Attribute:

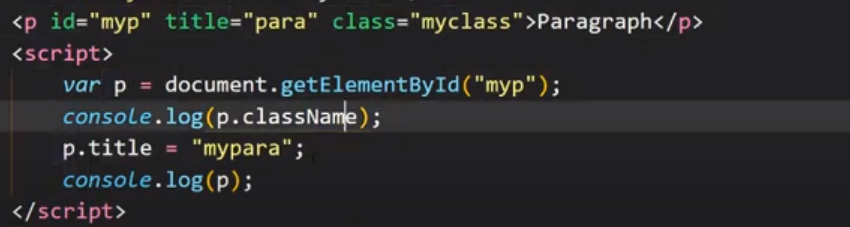




* HTML ATTRIBUTE Vs DOM PROPERTY:

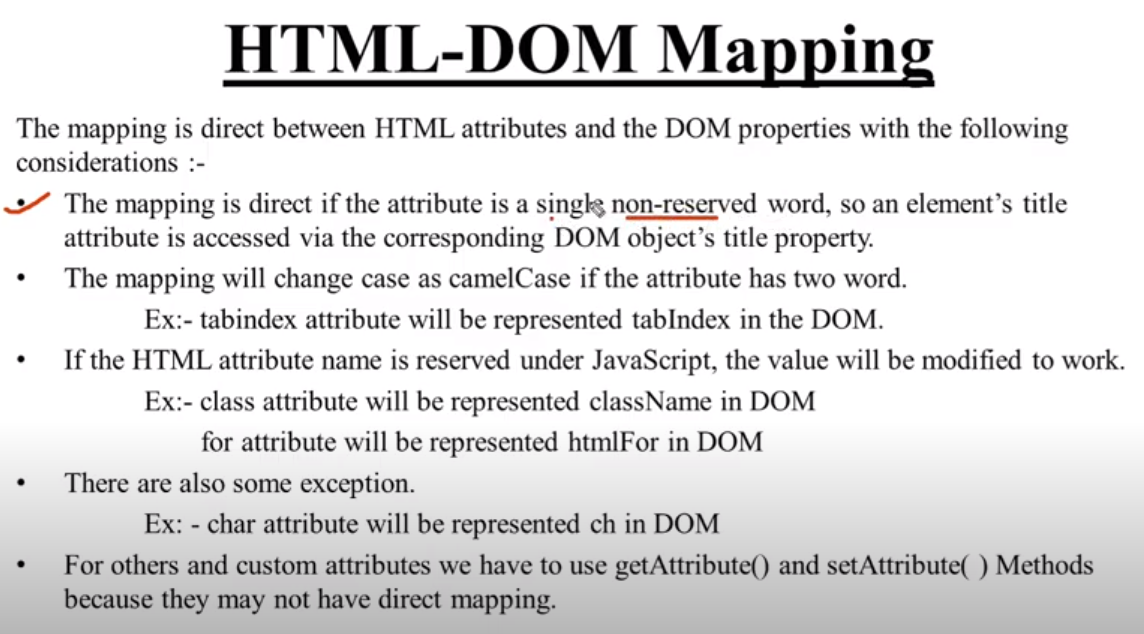


Like class is html attribute, but in dom, it is className and we used DOM in js, that’s why we used className in js



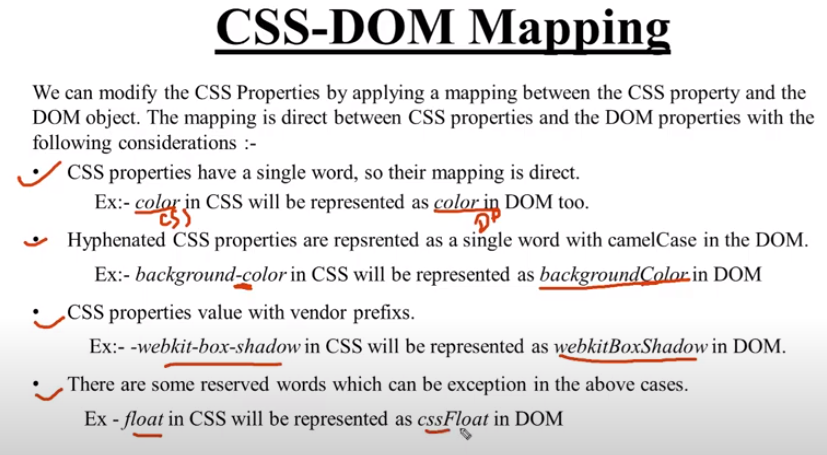
In script, we can write p.title or p.id but we cannot write p.class as it not supported in DOM, we have to write p.className

* HTML DOM MAPPING:

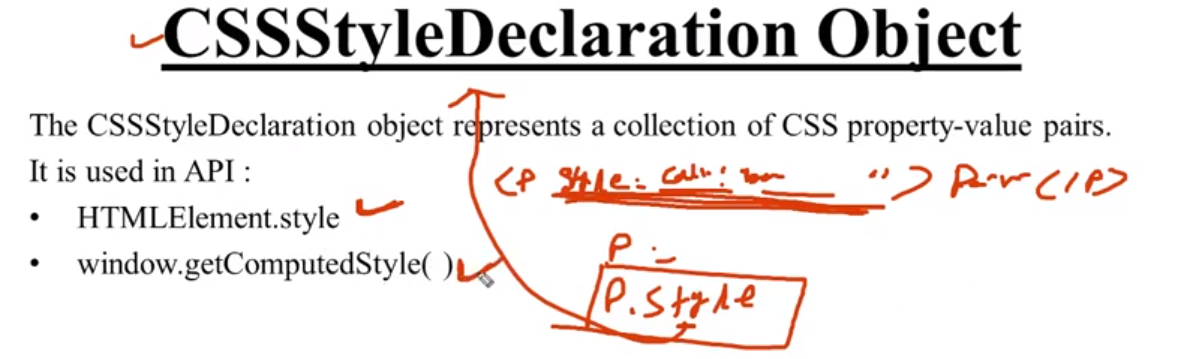


Example of first bullet -> document.getElementbyId(“id”).title or .class

* CSS DOM MAPPING:

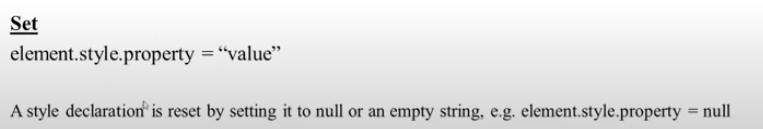


CSS Style Declaration Object:



It is only used with inline Css

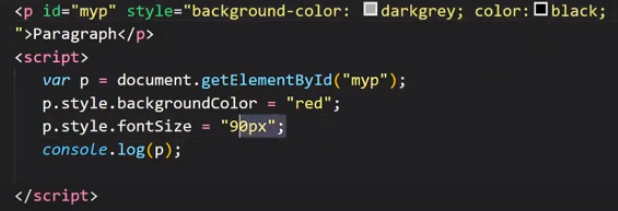
We used style property with element



Example:

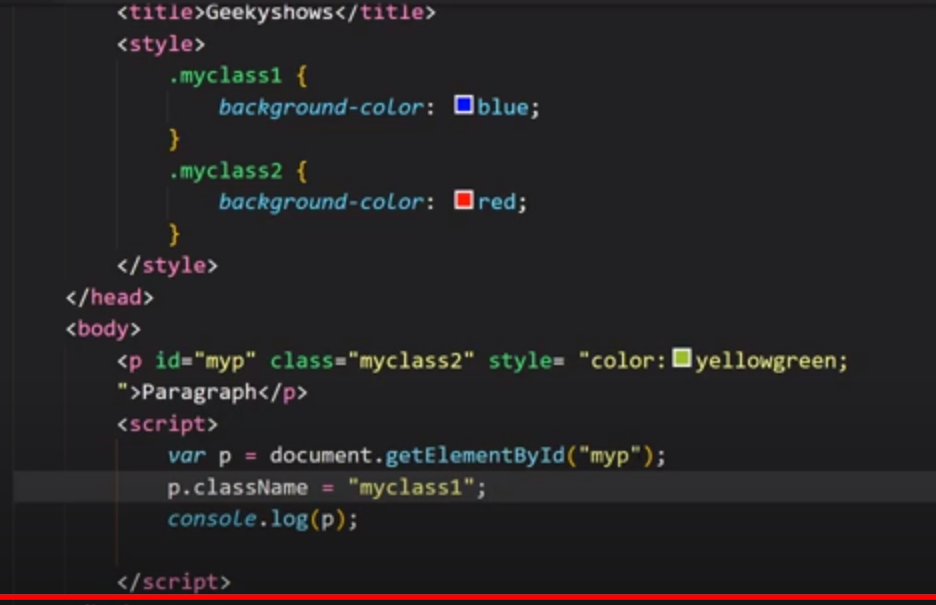


Remember, in js we use dom property instead of HTML attribute, that’s why used backgroundColor instead of background-color



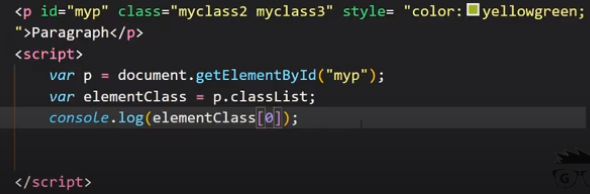
Hurdle in style property is that we can modify single style property at a time

DYNAMIC STYLE MANIPULATION:



* classList Property:

it returns token list containing length add remove etc for classes

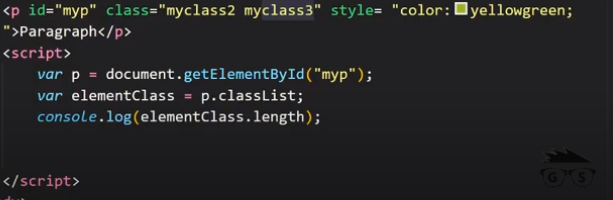


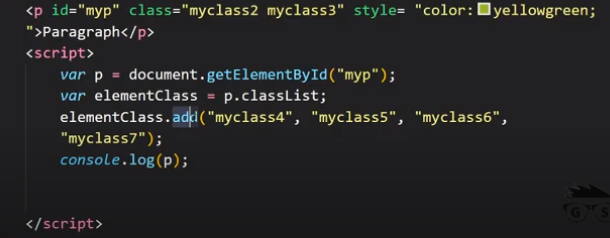
var elementClass = p.classList

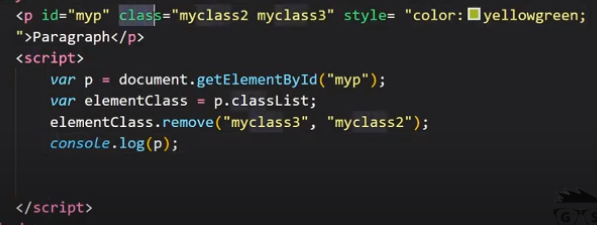
console.log(elementClass) -> returns list of classes used

now as per token list, we can apply many properties like to check no of classes used;

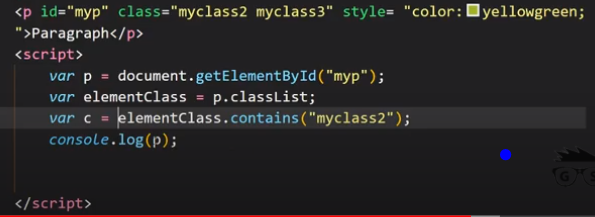
console.log(elementClass.length)

we can also add more classes using add

  
similarly we can also remove any classes

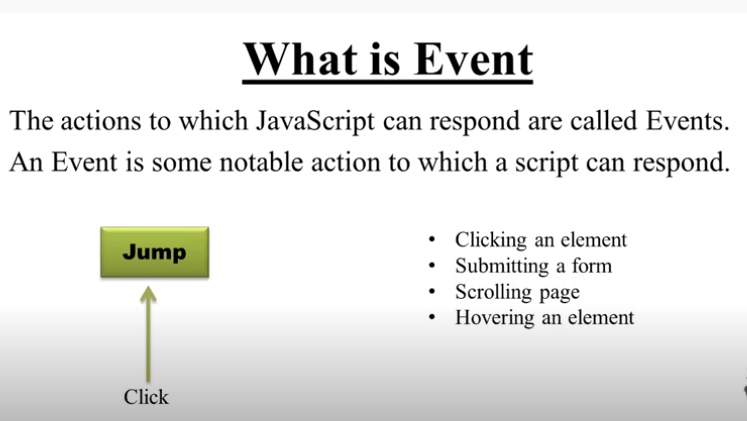


Checking whether class exist or not



From 82:

* Event



Event handler is a code of js which executed when given event occurs

Like when we click on button, event occurs so event handler code related to that event will be executed and between them is the event attribute like onClick onChange etc

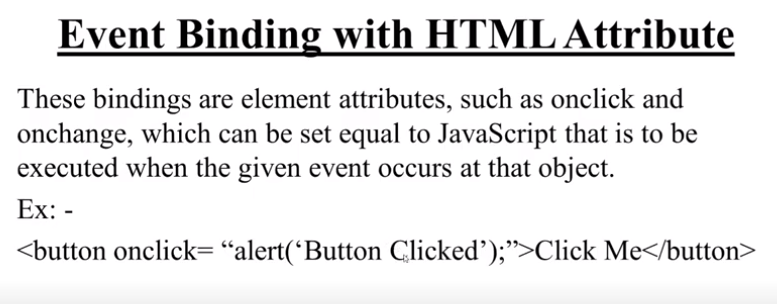
Like in js

<button onClick={(e)=>{ …code}}/> -> this is also called event binding with js

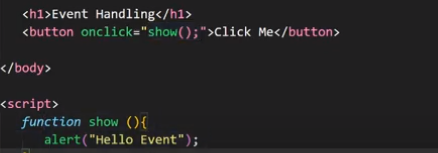
Here, onClick is the attribute

e is event

and logic {…code} is the event handler



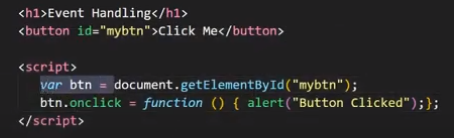
Example:



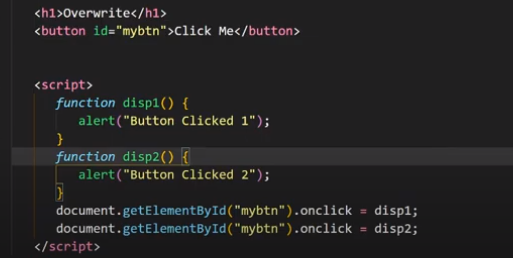
With event binding in html, we use “” and ; with onClick etc and we pass event attribute in html tag

* Event binding in JS:

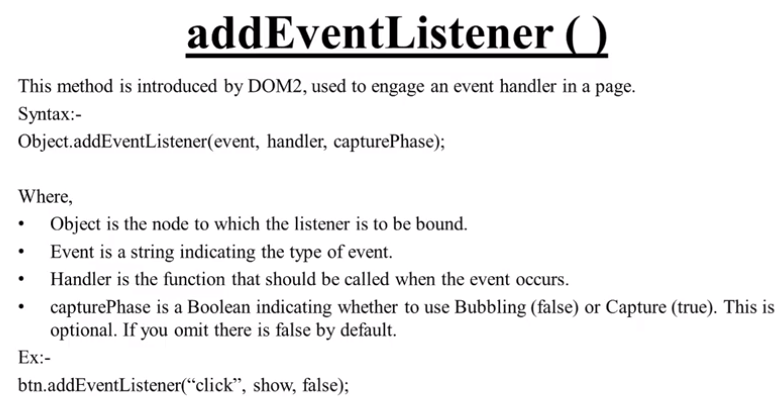
We create html tag, then we call it in js and assign event attribute to it but in js style

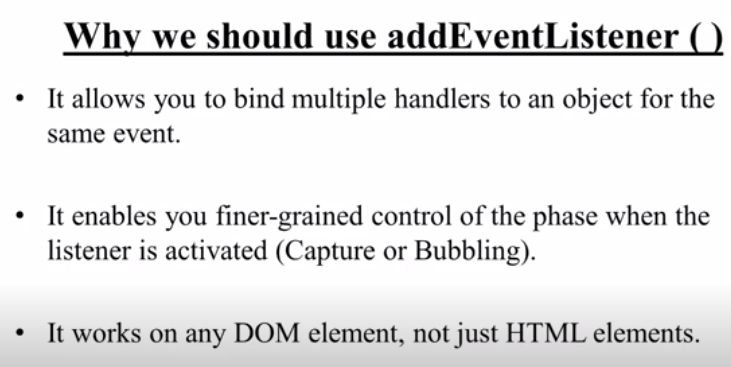


Overwriting Event handler:

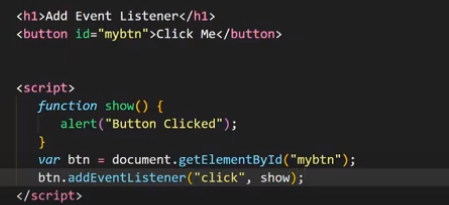


* addEventListener():

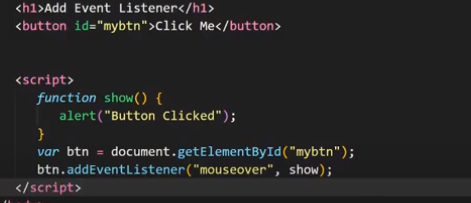




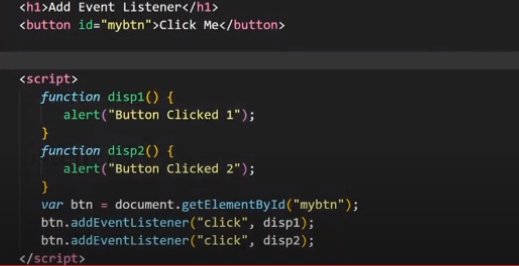
Example:



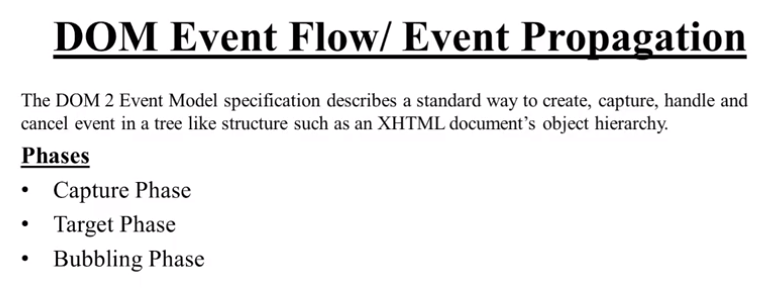
Similarly, another event is mouseover, when we move mouse over button, it show alert. The code is below



We can pass more than one handlers to an event through addEventListener



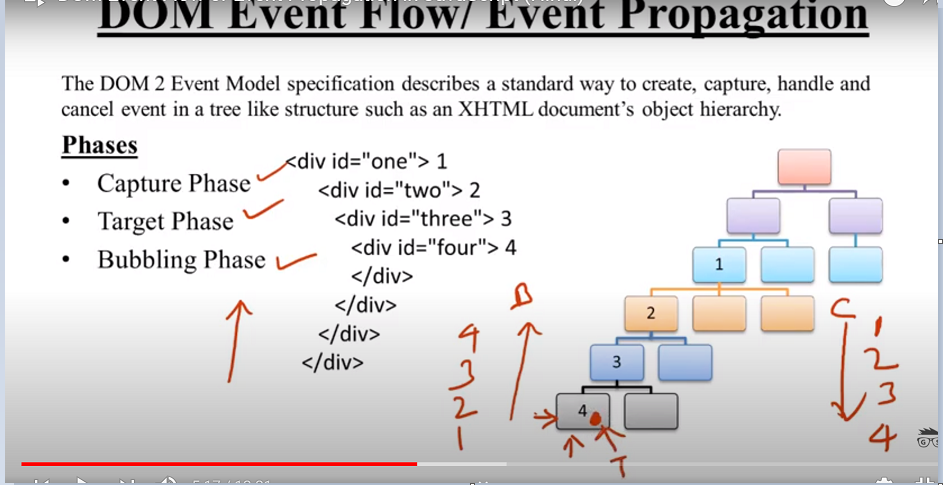
DOM EVENT PROPAGATION:



Capture -> top to bottom events

Bubbling -> bottom to top events

Target -> exactly same targeted events



Like if we have click event in 4 divs, then let save use click event wrt to div 4

Then in bubbling phase -> 4 then 3 then 2 and then 1

In capture phase -> It jumps to that div consist of click event which is 1 then 2 then 3 and then 4

And in target phase -> remains on 4

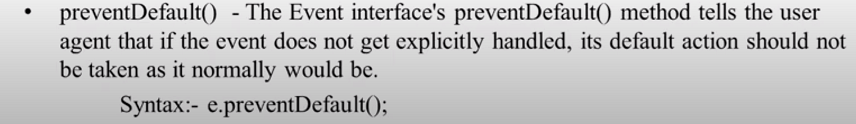
Bubbling phase -> element.addEventListener(‘even\_name’, function, false)

Capture phase -> element.addEventListener(‘even\_name’, function, true)

* Event object:

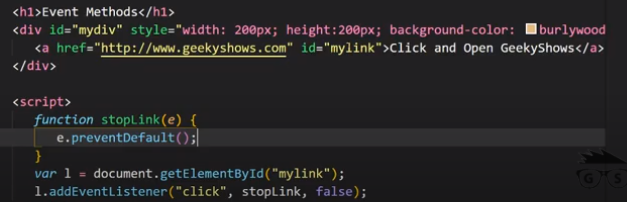


* e.preventDefault()



It prevents or stop from the default behavior of the event

Let say we apply click event on a link which redirect us to the content of that link. Let say for some time we don’t want to redirect, so we access that event and use addEventlistener and pass e.preventDefault to event of that element



Here when click on a tag link, it will redirect us to the website, but we call that a tag and using addEventListener on click prevent from its default behavior which is click, so thus we will not redirect to the side