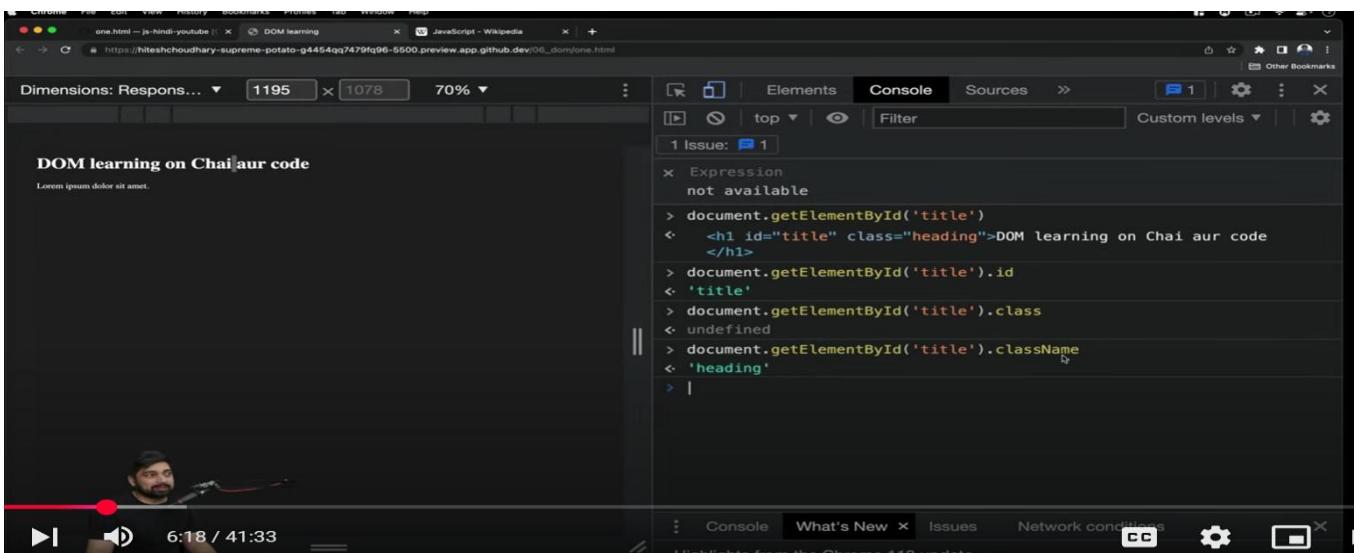
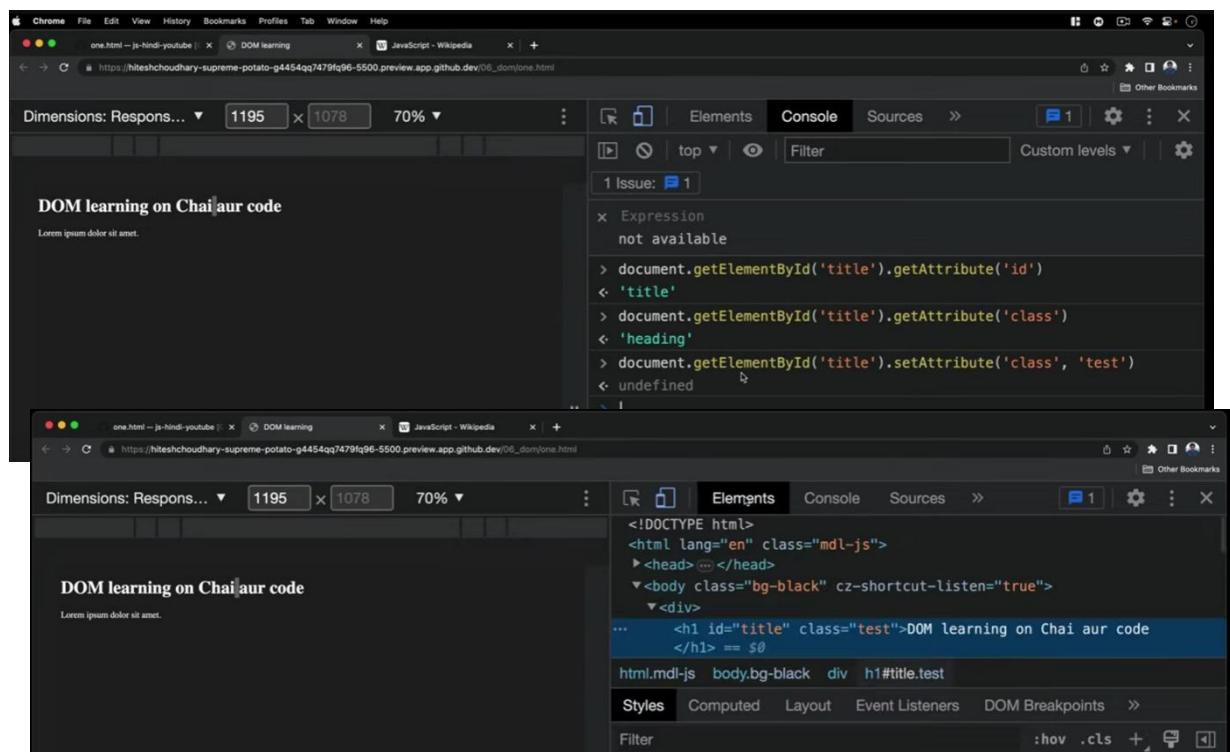


## DOM:- VS\_One.js file



Here we see that .class shows undefined but .class name shows 'heading' because although the class is known by there class name not simply by class only.



Here we see that while using setattribute we got undefined o/p but when we actually go to see the element in inspect there is change in class name  
ie. class name of h1 is heading but after using .setAttribute('class', 'test') it change to "test" as a class name.

DOM learning on Chai aur code

```

<!DOCTYPE html>
<html lang="en" class="mdl-js">
  <head> ... </head>
  <body class="bg-black" cz-shortcut-listen="true">
    <div>
      ... <h1 id="title" class="test heading">DOM learning on Chai aur code</h1>
    </div>
  </body>
</html>

```

h1#title.test.heading

Styles Computed Layout Event Listeners DOM Breakpoints

element.style {

Here again we want to set “test heading” as a class name. we see that while using .setAttribute we got undefined o/p but when we actually go to see the element in inspect there is change in class name ie. class name of h1 is heading but after using .setAttribute(‘class’ , ‘test’) it change to “test heading” as a class name.

```

> document.getElementById('title')
< <h1 id="title" class="test heading">DOM learning on Chai aur code</h1>
> const title = document.getElementById('title')

```

Here we use .getElemenByld(‘title’) to select the element by ID and for make futher manipulation easy on that element we store it into an variable name title (more detail we see in next snap)

```

DOM learning on Chai aur code
Lorem ipsum dolor sit amet.

> title
< <h1 id="title" class="test heading">DOM learning on Chai aur code</h1>
>

```

Here we see that when we simply write title and hit enter we get that assigned element easily as we previous stored in const title variable (previous snap)

## DOM learning on Chai aur code

Lorem ipsum dolor sit amet.

1 Issue: 1

```
x Expression  
not available  
> title.style.backgroundColor = 'green'  
< 'green'  
> title.style.padding = "15px"  
< '15px'  
> title.style.borderRadius = "5px"  
< '5px'  
> title.style.borderRadius = "15px"  
< '15px'  
>
```

**Now here we try to apply some style on that selected element which selected by id through title variable .**

**so when we apply style(bg-color , padding , border radius) on title that also refelect in main element which is selected by id .**

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>DOM</title>  
  
</head>  
<body style="background-color: black; color: white;">  
  
    <div>  
  
        <h1 id="title" class="Heading">This is chai aur code java script in hindi  
        <span style="display: none;">hidden text</span></h1>  
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Modi, eum.</p>  
  
    </div>  
</body>  
</html>
```

# This is chai aur code java script in hindi

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Modi, eum.

**Here is the snap of actual html code and with there corresponding output on which we further apply javascript syntax to manipulate it .  
here we see that the hidden text is not visible in output because it is not displayed by using css style property (hidden)**

This is chai aur code java  
script in hindi

Lorem ipsum dolor sit amet consectetur, adipisicing elit.  
Modi, eum.

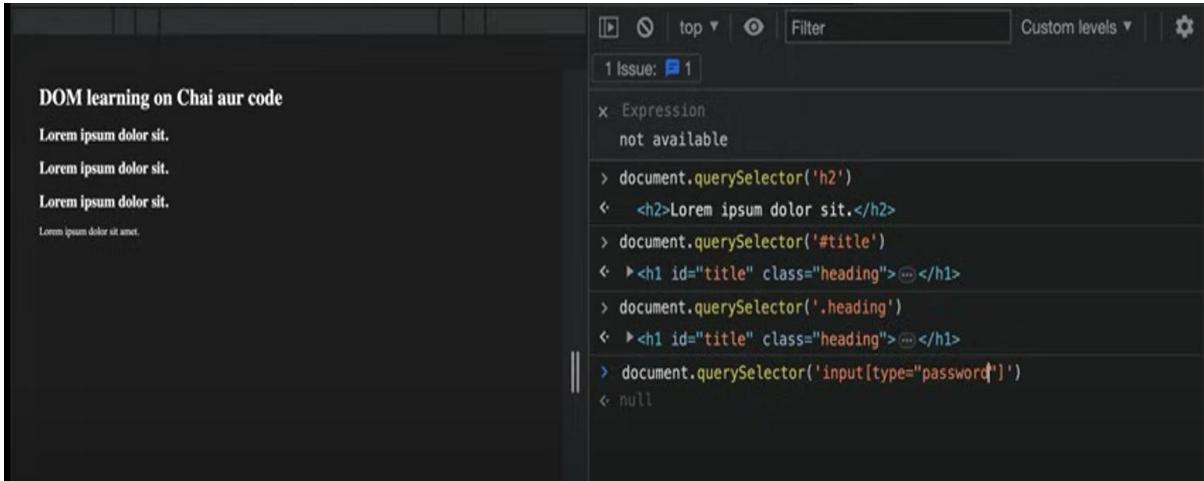
```
> title
< <h1 id="title" class="Heading">
    "This is chai aur code java script in hindi "
    <span style="display: none;">hidden text</span>
</h1>
> title.innerText
< 'This is chai aur code java script in hindi'
> title.textContent
< 'This is chai aur code java script in hindi hidden text'
> title.innerHTML
< 'This is chai aur code java script in hindi <span style="display: none;">hidden text</span>'
```

here we see that actual use of `.innerText`, `.textContent` and `.innerHTML`

- In `.innerText` we only get the text which is visible on actual document (such content we not retrive by this which is hidden by using css properties)
- In `.textContent` we get all the text present in selected element (which is visible or not visible in document) ie. It helps to show that hidden text along with all visible text also.
- In `.innerHTML` we get get all text with there html tags present in that element (in snap we se `<span>` tag also in o/p of `.innerHTML`)

this is the major difference b/t `innertext`, `textcontent` and `innerHTML`

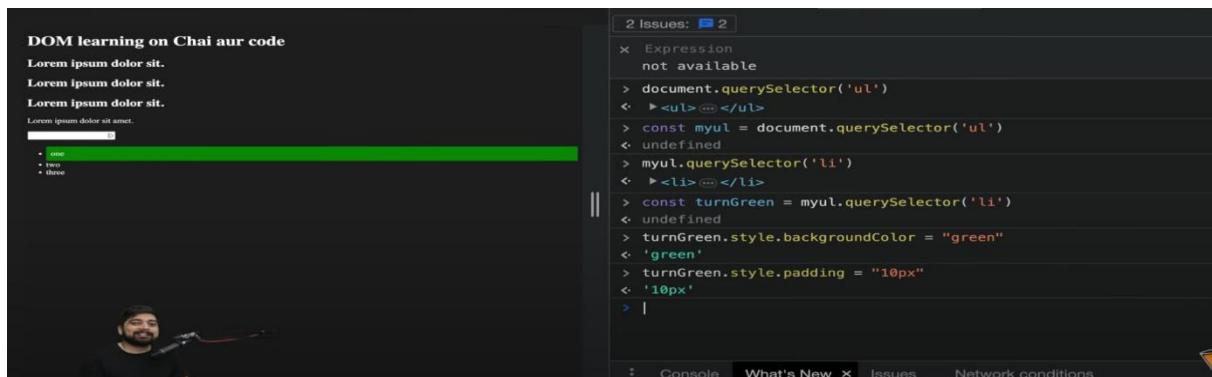
# Use of querySelector in dom manipulation



The screenshot shows the Chrome DevTools Issues panel with one issue. The issue is labeled "Expression not available". The code snippet below it shows several calls to `document.querySelector()` targeting various HTML elements like `<h2>`, `<h1 id="title" class="heading">`, and `<input type="password">`. The last line of the stack trace is `< null`.

```
1 Issue: 1
x Expression
not available
> document.querySelector('h2')
< <h2>Lorem ipsum dolor sit.</h2>
> document.querySelector('#title')
< ><h1 id="title" class="heading">...</h1>
> document.querySelector('.heading')
< ><h1 id="title" class="heading">...</h1>
> document.querySelector('input[type="password"]')
< null
```

Here we see that in `querySelector` use are able to use same syntax as in css to select the element , by title class and many more.

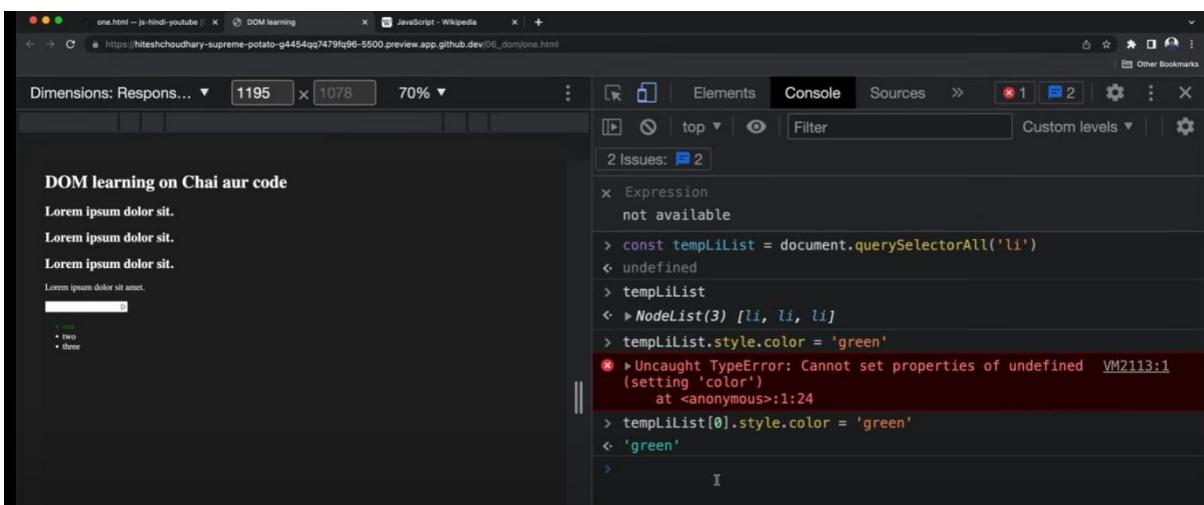


The screenshot shows the Chrome DevTools Issues panel with two issues. The first issue is "Expression not available" for `document.querySelector('ul')`. The second issue is "Expression not available" for `myul.querySelector('li')`. The code snippet shows a script block where `const myul = document.querySelector('ul');` is defined, followed by `myul.querySelector('li');`. The developer's video camera is visible in the bottom left corner.

```
2 Issues: 2
x Expression
not available
> document.querySelector('ul')
< ><ul> ... </ul>
> const myul = document.querySelector('ul')
< undefined
> myul.querySelector('li')
< ><li> ... </li>
> const turnGreen = myul.querySelector('li')
< undefined
> turnGreen.style.backgroundColor = "green"
< 'green'
> turnGreen.style.padding = "10px"
< '10px'
> |
```

Here we see that `queryselector` is used for list manipulation also

**querySelectorAll (Nodelist [A nodelist is a collection of document nodes which includes text nodes , attribute nodes and element nodes. A line break is also counted in nodelist but only first one. All subsequent line breaks are ignored by browser])** is different from array but somehow it behave like array , we can't direct apply changes on whole templist so we have to give the index of list element `tempList[0].style.color = 'green'`



The screenshot shows the Chrome DevTools Issues panel with two issues. The first issue is "Expression not available" for `tempLiList`. The second issue is an "Uncaught TypeError: Cannot set properties of undefined" at `VM2113:1`, which occurs when trying to set `tempLiList[0].style.color = 'green'`. The developer's video camera is visible in the bottom left corner.

```
2 Issues: 2
x Expression
not available
> const tempList = document.querySelectorAll('li')
< undefined
> tempList
< >NodeList(3) [li, li, li]
> tempList.style.color = 'green'
✖ > Uncaught TypeError: Cannot set properties of undefined VM2113:1
  (setting 'color')
    at <anonymous>:1:24
> tempList[0].style.color = 'green'
< 'green'
> |
```

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The title bar indicates 'Custom levels' and has a 'Filter' field. Below the title bar, there are two issues: 'Expression not available' and an 'Uncaught TypeError'. The error message is: 'Cannot set properties of undefined (setting 'color') at <anonymous>:1:18'. The code in the call stack is:

```
> const myH1 = document.querySelectorAll('h1')
< undefined
> myH1
<--> NodeList [h1#title.heading]
> myH1.style.color = 'green'
✖ Uncaught TypeError: Cannot set properties of undefined (setting 'color')
at <anonymous>:1:18
> myH1[0].style.color = 'green'
<-- 'green'
```

Here we see that the changement in color of h1 tag using querySelectorAll but there also we have to pass index as 0 b/c queryselectorall return an nodelist not an array or any single value.

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The title bar indicates 'Custom levels' and has a 'Filter' field. Below the title bar, there are two issues: 'Expression not available' and an 'Uncaught TypeError'. The error message is: 'Cannot set properties of undefined (setting 'color') at <anonymous>:1:18'. The code in the call stack is:

```
> tempLiList
<--> NodeList(3) [li, li, li]
> tempLiList.forEach(function (l) {})
< undefined
> tempLiList.forEach(function (l) {
    l.style.backgroundColor = 'green'
})
< undefined
> |
```

**Foreach loop on node list (here we can't use map or array functionality )so use foreach loop in node list is mandatory.** Even we can convert nodelist into array then onlt we can use all functionality of array on nodelist.

DOM learning on Chai aur code

```
2 Issues: 2
x Expression
not available
> document.getElementsByClassName('list-item')
<- > HTMLCollection(4) [li.list-item, li.list-item, li.list-item,
li.list-item]
> const tempclassList = document.getElementsByClassName('list-item')
<- undefined
> tempclassList
<- > HTMLCollection(4) [li.list-item, li.list-item, li.list-item,
li.list-item]
> tempclassList.forEach(function(li) {
  console.log(li)
})
✖ > Uncaught TypeError: tempclassList.forEach is not a
function
at <anonymous>:1:15
VM2926:1
> |
```

Here we see that how we select the element by ClassName in which the ClassName return the HTMLCollection Which is different from array and nodelist .

Here we are not able to apply loop (no any specific loop is defined to use in this htmlcollection) on HTMLCollection so we firstly convert the HTMLCollection into array.

DOM learning on Chai aur code

```
2 Issues: 2
x Expression
not available
> tempclassList
<- > HTMLCollection(4) [li.list-item, li.list-item, li.list-item,
li.list-item]
> Array.from(tempclassList)
<- > (4) [li.list-item, li.list-item, li.list-item, li.list-item]
> const myConvertedArray = Array.from(tempclassList)
<- undefined
> myConvertedArray
<- > (4) [li.list-item, li.list-item, li.list-item, li.list-item]
> |
```

Here we see that how an HTMLCollection is converted into array to apply loops and use all the useful feature of arrays.

DOM learning on Chai aur code

```
2 Issues: 2
x Expression
not available
> myConvertedArray.forEach(function(li){
  li.style.color = 'orange'
})
<- undefined
> |
```

Finally here we get an arrays properties in HTMLCollection and now applying loop (forEach ) on myConvertedArray.

**DOM:- VS\_Two.js file**