Exercise ! done see (mozilla & wikipedia for reference)

Javascript DOM (Document object model)

⇒ It is basically javascript object and access easily with console of browser

⇒ DOM is created by browser at ~~that~~ time of web page open

⇒ we can interact with html page with DOM

⇒ DOM is tree like structure and also has each & every info about ~~tags~~ elements, URL object with it.

⇒ DOM also showcase events on a particular element

to know about dom attribute

Imp `console.dir(document);` ⇒ this command will provide all info about a element

m= document.body
n= m.firstchild
m.firstElementChild
n.innerText
n.innerHTML

Command to try

to select element
querySelectorALL("".className")
↑ in selector we can pass #id, tag or .class

.style Dom operator can manipulate styling

selector → tag → "hi"
→ class → ".className"
→ id → "#idName"

for query selector

for any tag if you want info do `console.dir(m)` ⇒ such like that we can grab info

m= document.querySelector(".class")

⇒ Say change color of a element
m.style.color = "red" ⇒ this will change color

⊙ getElementById("#id") → here you have write name only not like selector

⊙ getElementByClassName("className") → just enter class name
↓ this will give list o/p ⇒ list

⊙ getElementByTagName("h1") ⇒ so return all h1 in file
↳ o/p ⇒ list ⇒ can use array indexing to access single (one)

⊙ querySelector("selector") → pass selectors like we use in css file

⊙ querySelectorAll("selector") ⇒ If multiple values are there for some selector so it will return list related to selector

.className → #idName → tagName → only return first one

Element text manipulation

treat internal tag text as html

textContent ⇒ this will give text in that tag
innerHtml ⇒ this will give info inside the html tag
outerHtml ⇒ this will give whole entire html element
↳ that is <h1> innerText </h1>

→ only inside text | If you pass html in innerText it treated as text

⇒ innerText ⇒ ~~gives~~ out text inside a tag
↳ It excludes all html tags present inside

not exclude HTML tags

→ textContent will give text also also give info about newline

---

⇒ If we want to change html tags of own element
say h1 → h2

```
m = document.querySelector("h1")
m.outerHtml = "<h2> content </h2>"
```

h1 → Converted to h2

---

change class Dom

```
Var el = document.querySelector("h1");
Console.dir(el)
```
→ show info about tags

to add a class in HTML Element

`S = document.querySelector("p")`

`S.classlist` ⟹ this will gives us info about classes present for that tag

`S.classlist.add("class to add")` ⇒ to add a class in element

`S.classlist.remove("className to remove")`

↳ this will remove class from the html tag/element

or `S.className = "classes seprated by comma"`

Say you want to add class1 class2 class3

`S.className = "class1 class2 class3"` ← that you have to do

`S.classlist.toggle("classname")`

If class exist will remove it & not exist in it will add it (shuffle classname in & out)

#change style Name

⇒ .style for selected element is done in order to do style for it and it use javascript to modify styles in ~~java~~ html

⇒ there are allot of style specification are these and you can use them ~~inside~~ with DOM

```
Var m = document.querySelector("h1")
m.style.color = "red" → output allwn should be in string
```

⇒ Dom overwrite the previously applied styling which is main task to do i.e. to overwrite previous styles

---

⊕ Element attribute manipulation

so each element contains attribute so using Dom we can change internal attribute

`<a href="link" > name </a>` → attribute

So let grab a <a>tag

let m = document.getElemantByTagName("a")

m.attribute gives attribute related to that tag

m.attributes is list so m.attribute[0].value

this will give value of attribute

and you can also assign value with

m.attribute[0].value = "something"

⊙ getAttribute("name") ⇒ function cannd used to get attribute (value) of for a tag

o/p:) '#' ⇒ value

⊙ setAttribute("Name of attribute", "value") → string

thus how can you get & set attribute value

⊞ Event listener

this as it is used to get to Elemnt by which event is arised

var m = document.queryselector("ul")

m.addEventListener("click", function() {

console.log(this)

}

→ this will print the result

⇒ so you can get this any manipulate it

---

this ⇒ it is one which give info about what triggered a event (Imp)

⊞ Keypress event

query in Dom say a tag is input & has name: some

document.queryselector("input[name = "$some"]");

such like that we can be more precise

there are large no. of events are there in JS which we can can adjust in addevent listner

Ex: click, keypress, & many more

do a search

⇒ $event ⇒ this is event parameters related to task that ansed ~~on~~ for triggering event

```
a.addEventListner("click", function($event){
    console.log($event);
```

⊙ this ⇒ It is the tag on which event invoked → Element

$event.type ⇒ the type of event

$event.key ⇒ info about value which invoked event

⇒ for enter keycode == 13

→ ~~invol~~ change paragraph

JS mouse Events

do a search & read w3school article on mouse events

⊙ mouseover (when you how you can do anything)

⇒ we can create multiple event listner on single element

→ to and it mouseout

⇒ with javascript you have to use (mouseover + mouseout) to create hover effect like pseudo classes in CSS

Create own elements

use ⇒ document.createElement("tagName")

with this you can able to create own tag in JS & Dom

⇒ appendchild ⇒ so say have list of tags so it will add Elem at last.

⇒ m = document.createElement("tagName") → Say hi

· m.innerText = "Hello world"

our new block is ready

K = document.querySelector("target value")

K.appendChild(m) → strictly Pass a tag

Such like that we can create and add new elements also you can add event listener to them.

Project dynamic list

.ParentElement ⇒ this will give location of parent which we can use to do things