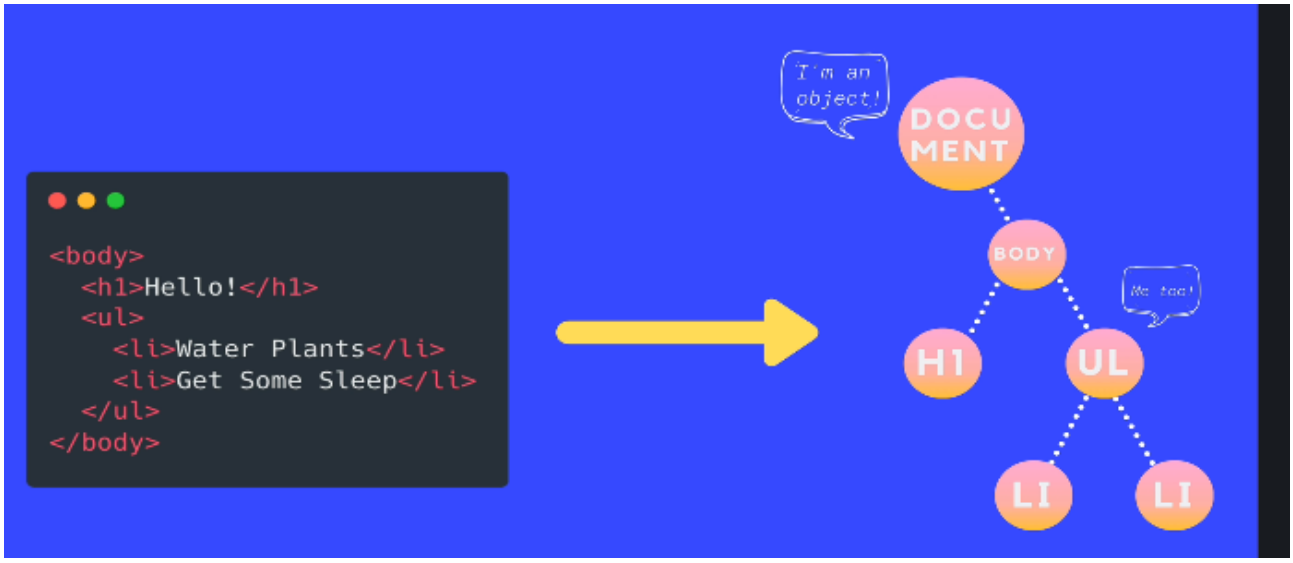
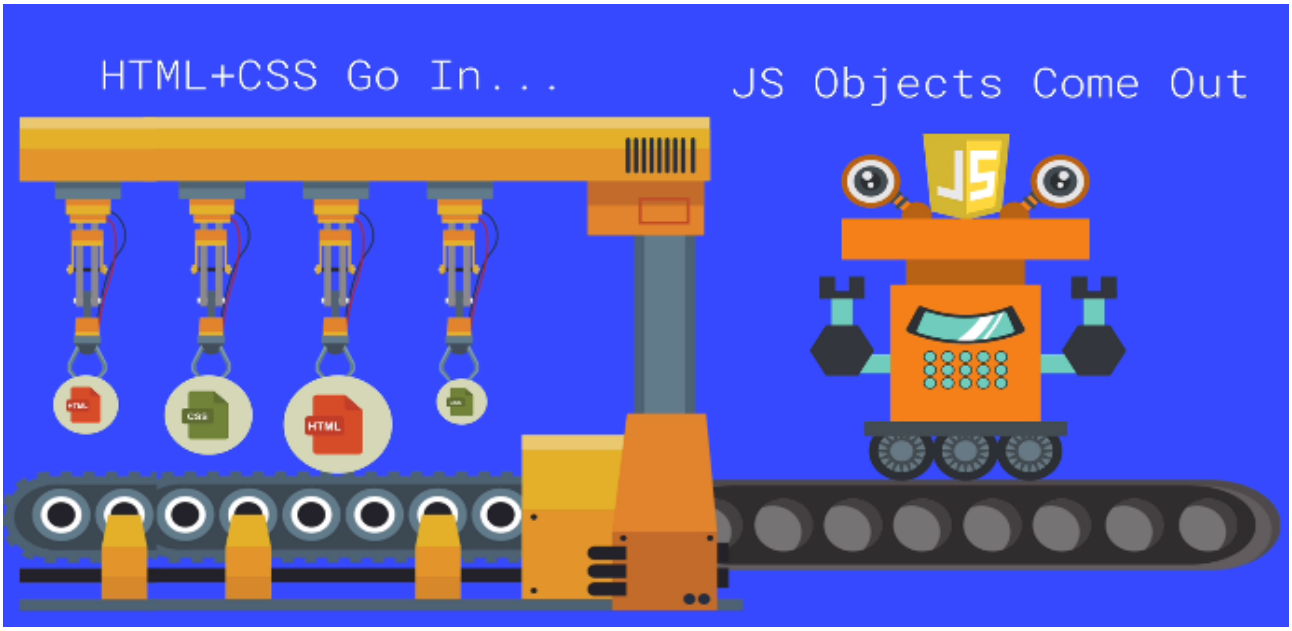
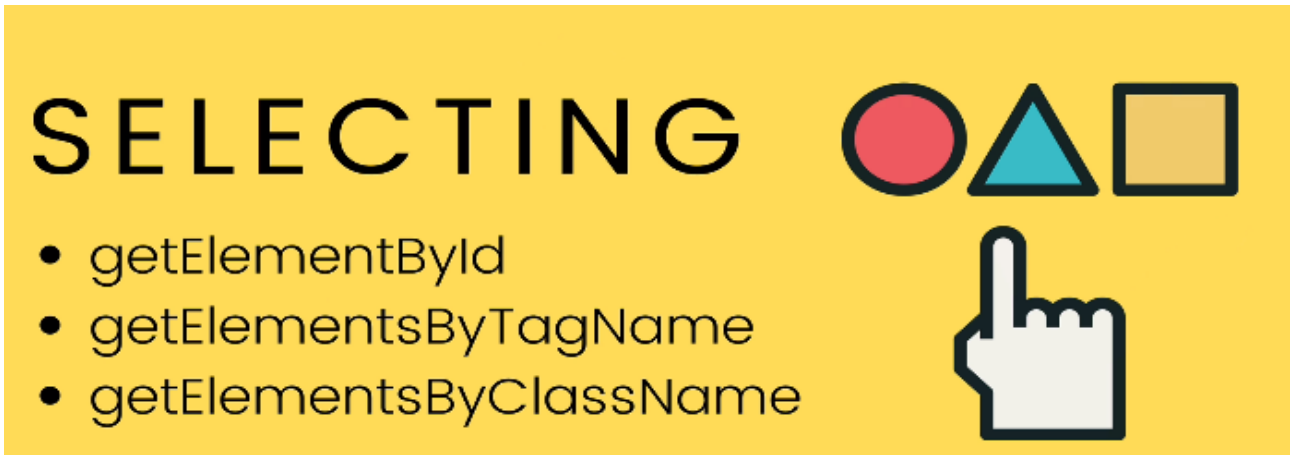
WHAT IS IT?

- The DOM is a JavaScript representation of a webpage.
- It's your JS "window" into the contents of a webpage
- It's just a bunch of objects that you can interact with via JS.





```
> document
<- ▼#document
      <!DOCTYPE html>
      <html dir="ltr" lang="en" class="focus-outline-visible">
       ▶<head>…</head>
       ▶<body style="background-color: rgb(255, 255, 255);">…</body>
      </html>
> console.dir(document)

  ▼#document i
     URL: "chrome://new-tab-page/"
    ▶activeElement: body
    ▶adoptedStyleSheets: []
     alinkColor: ""
    ▶all: HTMLAllCollection(144) [html.focus-outline-visible, head, meta, title, style, custom-styl
    ▶anchors: HTMLCollection []
    ▶applets: HTMLCollection []
     baseURI: "chrome://new-tab-page/"
     bgColor: ""
```



Like
document
>>> gives u the html
so u have to do
console.dir(document)
>>> to get the object

Likewise
whenever you use select elements using Dom, you have to use console.dir to view the object.

It’s iterable but itsn’t an array so u can’t use map,reduce..

In query selector, we pass the “sameway of selector we pass in Css” queryselector only selects one elements and return.

innerHtml , textContent & InnerText

textContent: is gonna look like the how u written your html content.even though any is display:none , You can see in the textContent.
So every piece of content is gonna show.
innerHtml:what you see on the browser exactly like that. It is sensitive to what is showing on the moment .

e.g
const x=document.querySelector(“nav a”);
x.innerText=” mumbai_foodbasket ”
<!-- this will not work, it will consider b tag as string.
//so use
x.innerHtml=” mumbai_foodbasket ”

innerHtml	textContent	innerText
smart	better	compact
	draw as written in .html	dumb
	Not dumb, sensitive to what is currently shown.	

Attributes
---
const firstlink=document.querySelector(“a”).src
firstlink.setAttribute(‘href’,’https://www.udemy.com/course/the-web-developer-bootcamp/learn/lecture/22003782#overview’)

```
document.querySelector('a').title
"List of chicken breeds"
const firstLink = document.querySelector('a')
undefined
firstLink.href
"file:///wiki/List_of_chicken_breeds"
firstLink.getAttribute('href')
"/wiki/List_of_chicken_breeds"
```

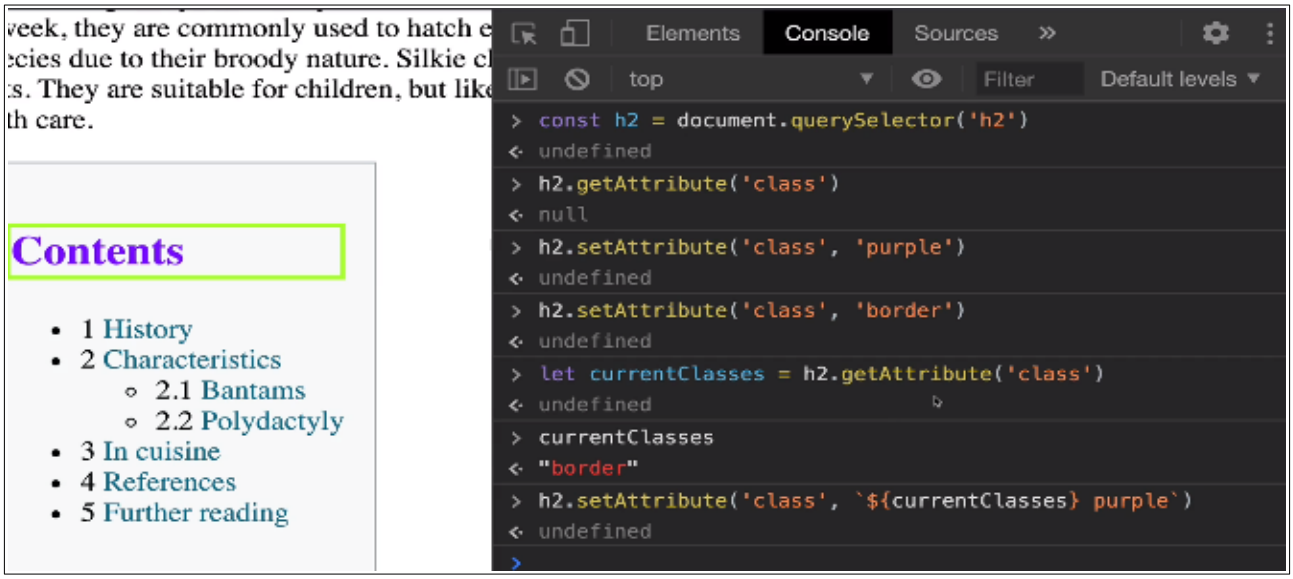
when you use .*getAttribute*() you are getting directly from the html itself. & when you access property directly on the element like “.href” i.e is going through js. So we can see we got different value.

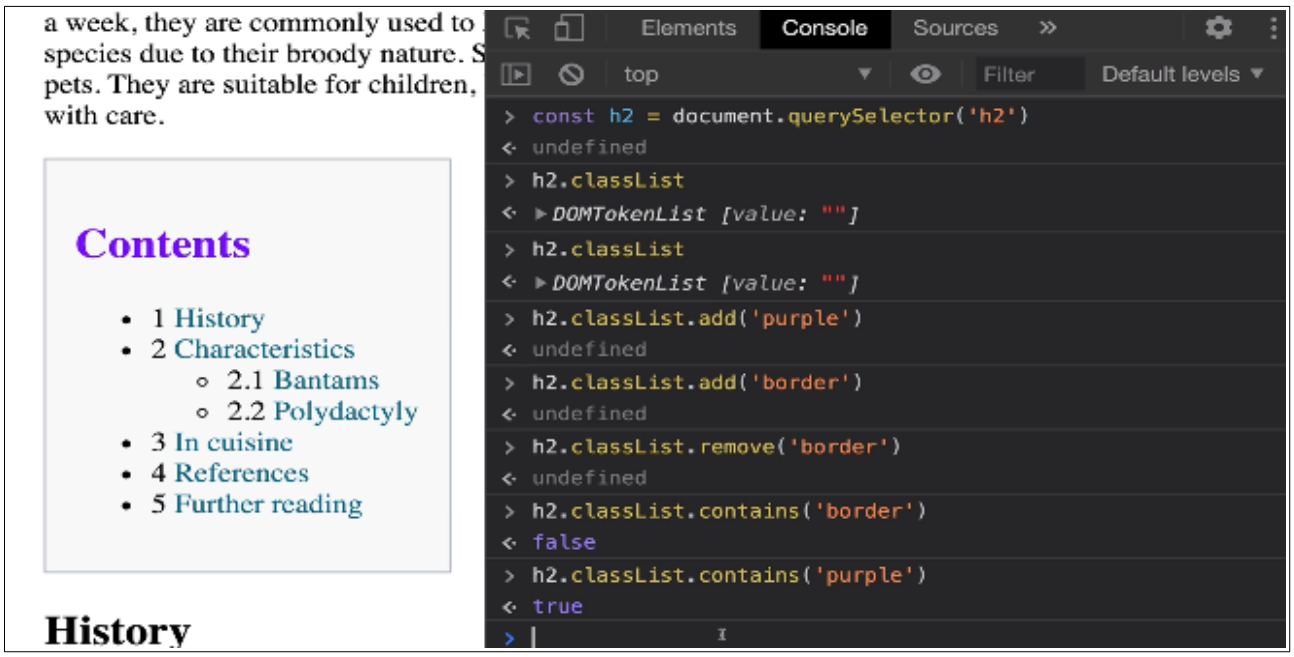
Styles
---
selected element object has a property i.e style
h1.style <!-- In Css we have background-color, but in the style object we dont have “-”. Instead its camelcase i.e backgroundColor
h1.style <!-- only stores inline style, not from our stylesheet

Here, you whatever you do would be inline style, you should avoid using inline style.
i.e h1.style.color=blue;

-Better way to apply styles to elements is to use a class
Issues with style property:
-inlines are bad
-using style property you can only see style attribute.
So unless you wrote the style inline, you can’t read the style.I cant tell what is the fontSize of the element...
So what if i wanna make something the same font-size idk how big it is, atleast by not looking at my css

So there is another way. To find the actual computed style once everything is applied. It’s not as easy just to look at your .css copying the style over. Because you can have multiple stylesheet n they may have conflicting sytles and specificity.
-You have to wait until everything is loaded and computed to figure out actual style.you can do by using
window.getComputedStyle(h1).fontSize
This is not a query selector, this is the actual element object in dom






```
h2.classList.toggle('purple')
true
```

It might seems easier to change style using style property but by classlist its much more simple where u can apply bunch of property by adding class or multiple classes.You can also remove class and toggle.

Traversing parent/child/sibling

---

i.e

parentElement
Children
nextElementSibling
previousElementSibling

Last child
LastElementChild
NextSibling

All this property allows us to navigate,traverse,move from one element to some relative or to it's parent or to it's parent parent.

```
firstBold.parentElement
 ▶<p>…</p>
firstBold.parentElement.parentElement
 ▶<body>…</body>
firstBold.parentElement.parentElement.parentElement
  <html lang="en">
   ▶<head>…</head>
   ▶<body>…</body>
  </html>
const paragraph = firstBold.parentElement
undefined
paragraph.children
▶HTMLCollection(8) [b, b, a, a, a, a, a, a]
paragraph.children[0]
  <b>Silkie</b>
paragraph.children[0].parentElement
 ▶<p>…</p>
```

nextSibling and previousSibling is gonna give us a node.
For e.g inside <h1>...</h1> we have a text node.
Somebrowser make new lines a text node.

createElement

const newImg=document.createElement(‘img’)
console.dir(‘newImg’) <!--to view
newImg.src=”<someurl>”
document.body.appendChild(newImg)

```
const newH3 = document.createElement('h3')
undefined
newH3
  <h3></h3>
newH3.innerText = 'I am new!'
"I am new!"
document.body.appendChild(newH3)
  <h3>I am new!</h3>
```

appendChild

```
> const p = document.querySelector('p')
<- undefined
> p.append('i am new text yaaaaaayyy!!!')
<- undefined
> p.appendChild('i am new text yaaaaaayyy!!!')
✕ ▶Uncaught TypeError: Failed to execute 'appendChild' on 'Node': parameter 1 is not of type 'Node'.
    at <anonymous>:1:3
>
```

```
p.append('i am new text yaaaaaayyy!!!', 'asdasdasdasdasdasd')
```

It allow us to insert more than one thing at a time

```
const newB = document.createElement('b')
undefined
newB.append('Hi!')
undefined
newB
  <b>Hi!</b>
p.prepend(newB)
undefined
```

y.append(div1,div2)

Element.insertAdjacentElement [click]

Parameters

position

A `DOMString` representing the position relative to the `element`; must be one of the following strings:

- `'beforebegin'`: Before the `element` itself.
- `'afterbegin'`: Just inside the `element`, before its first child.
- `'beforeend'`: Just inside the `element`, after its last child.
- `'afterend'`: After the `element` itself.

```
> const h2 = document.createElement('h2')
<- undefined
> h2.append("Are adorable chickens")
<- undefined
> h2
<-   <h2>Are adorable chickens</h2>
> const h1 = document.querySelector('h1')
<- undefined
> h1.insertAdjacentElement('afterend', h2)
<-   <h2>Are adorable chickens</h2>
> h1.nextElementSibling
<-   <h2>Are adorable chickens</h2>
```

```
const h3 = document.createElement('h3')
undefined
h3.innerText = 'I am h3';
"I am h3"
h1.after(h3)
undefined
```

```
const firstLi = document.querySelector('li')
undefined
firstLi
▶<li class="toclevel-1 tocsection-1">…</li>
const ul = firstLi.parentElement
undefined
ul
▶<ul>…</ul>
ul.removeChild(firstLi)
▶<li class="toclevel-1 tocsection-1">…</li>
```

Another way

firstLi.parentElement.removeChild(firstLi)

<!-- This works but isn't considered best practice

Another way

```
const img = document.querySelector('img')
undefined
img.remove()
undefined
```

Inline event
for e.g
<button onclick=”alert(‘you clicked a button’)”>click Me </button>
If you wanna write multiple lines in you have to make seperation using “;” i.e
just like in linux cmd
for double click
<button ondblclick=”alert(‘you clicked a button’) ;alert(‘stop clicking me’)”>click Me </button>

```
onblur: null
oncancel: null
oncanplay: null
oncanplaythrough: null
onchange: null
onclick: null
onclose: null
oncontentvisibilityautostatechange:
oncontextlost: null
oncontextmenu: null
oncontextrestored: null
oncopy: null
oncuechange: null
oncut: null
ondblclick: null
ondrag: null
ondragend: null
ondragenter: null
ondragleave: null
ondragover: null
ondragstart: null
ondrop: null
ondurationchange: null
onemptied: null
onended: null
onerror: null
onfocus: null
onformdata: null
onfullscreenchange: null
onfullscreenerror: null
ongotpointercapture: null
oninput: null
oninvalid: null
onkeydown: null
onkeypress: null
onkeyup: null
onload: null
onloadeddata: null
onloadedmetadata: null
onloadstart: null
onmousedown: null
onmouseenter: null
onmouseleave: null
onmousemove: null
onmouseout: null
onmouseover: null
onmouseup: null
onmousewheel: null
onpaste: null
onpause: null
onplay: null
onplaying: null
onpointercancel: null
onpointerdown: null
onpointerenter: null
onpointerleave: null
onpointermove: null
onpointerout: null
onpointerover: null
onpointerrawupdate: null
onpointerup: null
onprogress: null
onratechange: null
onreset: null
onresize: null
onscroll: null
onsearch: null
onsecuritypolicyviolation:
onseeked: null
onseeking: null
onselect: null
onselectionchange: null
onselectstart: null
onslotchange: null
onstalled: null
onsubmit: null
onsuspend: null
ontimeupdate: null
ontoggle: null
ontransitioncancel: null
```

```
> Gon=document.querySelector("a[data-tool-tip]")
<- ▶<a href class="dates" data-tool-tip="My name is number1">…
   </a>
> console.dir(Gon.innerText)
   Gon                                                VM711:1
<- undefined
> console.dir(Gon)
                                                      VM749:1
   ▼a.dates i
      accessKey: ""
      ariaAtomic: null
      ariaAutoComplete: null
      ariaBrailleLabel: null
      ariaBrailleRoleDescription: null
```

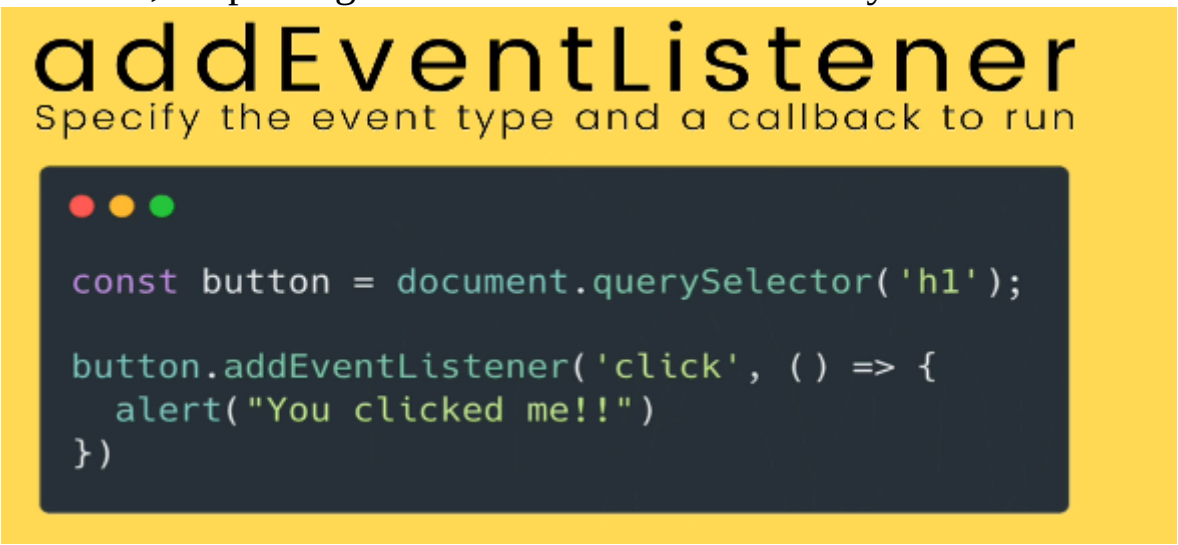
So basically dom objects have properties for e.g onclick or onmouseenter we can assign a function to it, so whenever there is a click or mouseenter it will run that function

A very important note

Set of do and donts
`` mydiv1.onclick=alert("hello there sahil"); ``` this isn’t equivalent to ``` mydiv1.onclick=function(){   alert("hello there kazim") } ```
Your properties should be a given a proper function. The mistake is that it will run the function right away. Whereas the below one. We are setting a property to a function. It should refer to a func.

```
btn.onclick = function () {
    console.log("YOU CLICKED ME!")
    console.log("I HOPE IT WORKED!!")
}

function scream() {
    console.log("AAAAAHHHHH");
    console.log("STOP TOUCHING ME!")
}

btn.onmouseenter = scream;
```

See Above, u r passing the function on the event but you aren't executing.



Not all events works on every elemnent . For e.g keypress doesn't work on a button.

You wanna use addEventListener over onclick/..... cuz you can't have two callback functions on the same event for e.g click.

mybtn.onclick=sayMyName; Doesnt work
mybtn.onclick=sayMyNickName; ✗ only listen for last event
---
mybtn.addEventListener('click',sayMyName) ✓ Here,
mybtn.addEventListener('click',sayMyNickName) both events are listening

```
tasButton.addEventListener('click', twist, { once: true })
tasButton.addEventListener('click', shout)
```

Tip: when you set property
document.body.backgroundColor='rgb(200,255,300)'; <!-- key:value(str)

Events and the keyword this(Important 3stars)

I think we are not giving enough credit to this.

Summary:The keyword this with a callback

Aim:

Reasoning:Basically on click events we want the text and the background-color to change.

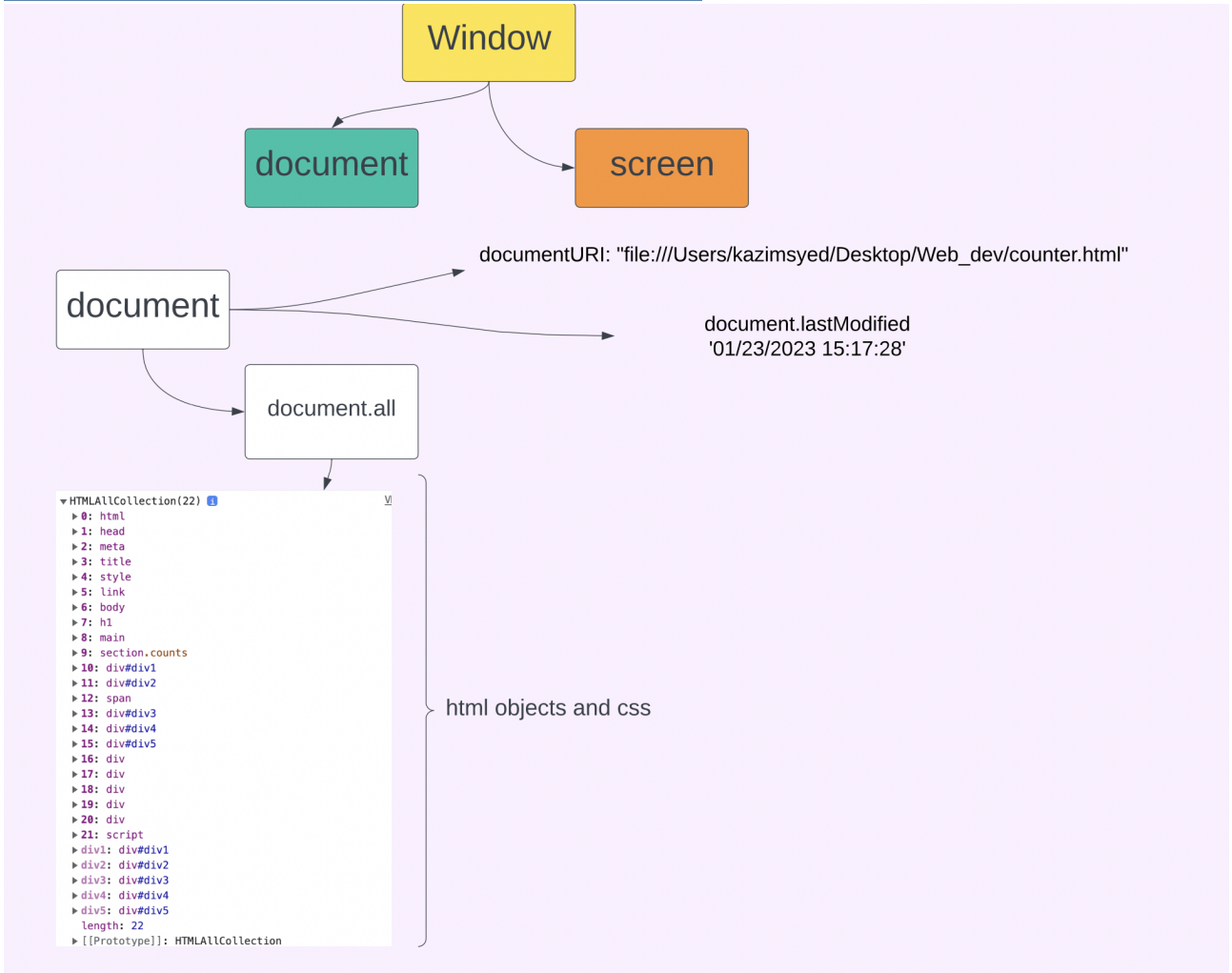
We were having the same code written twice. So we used a function.
A function when passed into like an argument is a callback.
The Keyword this in this function aka callback refers to the dom object on which the event was fired.

```
const buttons = document.querySelectorAll('button');

for (let button of buttons) {
    button.addEventListener('click', function () {
        button.style.backgroundColor = makeRandColor();
    })
}
```

```
for (let button of buttons) {
    button.addEventListener('click', colorize)
}

const h1s = document.querySelectorAll('h1');
for (let h1 of h1s) {
    h1.addEventListener('click', colorize)
}

function colorize() {
    this.style.backgroundColor = makeRandColor();
    this.style.color = makeRandColor();
}
```

The keyword ‘this’ when You have it inside a callback ,that is invoked by some event handler, ‘this’ refers to the element the eventlistener was listening.

Keyboard Events and Event Objects



main > Section.counts

TagName:”SECTION”

ParentNode:main

ChildNodes:NodeList(11) [text, div, text, div, text, div, text, div, text,…]

Children:HTMLCollection(5)[div,div,div,div,div]

firstChild:text

firstElementChild:div

classList > className:”counts”

previousElementSibling:null
previousSibling:null

lastchild:text
lastElementChild:Div

.attributes.length
.attributes['class'].nodeName

.innerHtml() : just gives me the the html with the content. Boring

.innertext() : gives me only the text that is visible to the user within the tags. Amazing+Compact

textContent(): gives me the text within the tags

style

main > Section.counts>div

TagName:"DIV"

style

ParentNode:section.counts

ChildNodes:NodeList(1) [text]

Children:HTMLCollection()[]

firstChild:my name is sahil1

firstElementChild:""

classList > className:""

previousElementSibling:null
previousSibling:text

nextElementSibling:div
previousSibling:text

.attributes.length :2
.attributes['data-tool-tip'].nodeName
.attributes['id'].nodeName
e.attributes['data-tool-tip'].nodeValue
'my name is sahil11'

.innerHtml() : just gives me the the html with the content. Boring

.innertext() : gives me only the text that is visible to the user within the tags. Amazing+Compact

textContent(): gives me the text within the tags

diff between KeyUp and KeyDown	In the case of KeyUP,if u hold the key down for a while and then release for e.g backspace. It will fire only one event for KeyUp. But for the case of keydown, if i am holding backspace, it will register backspaces. Imagine i wanna delete the entire line at my cursor is at the end. so i would hold it.
Similarities for KeyUp and KeyDown	Keyboard events are only generated by <input>, <textarea>, <summary> and anything with the contentEditable or tabindex attribute.
Keyboard events	They can be fire for like input[type='text] when im hitting the down array key. There isnt any change in the input. But the event is getting triggered.

```
> document.addEventListener('keydown',(e)=>{
      console.log(e.code,e.key)
  });
<· undefined
  KeyA a
  KeyB b
  CapsLock CapsLock
  ShiftLeft Shift
```

Figure 1: e.code , e.key

what i am writing is are the key and the code is the codified name	keyboard events(e.code,e.key)

If u change ur language settings, so where your is w,a,s,d or u may not even have those letters aka key but u care about the position of w,a,s,d for ur game. So use code.
But if i use ‘code’ it doesn’t matter. So if we care about position on keyboard use ‘code’. But what you care about is the ‘char’, use ‘key’.
For e.g -y for yes and its a weird keyboard.
Again, if you dont care which button you pressed to generate the char use ‘key’ then.

Keyloggers

Use window.Eventlistener()

```
window.addEventListener('keydown', function (e) {
    switch (e.code) {
        case 'ArrowUp':
            console.log("UP!");
            break;
        case 'ArrowDown':
            console.log("DOWN!");
            break;
        case 'ArrowLeft':
            console.log("LEFT!");
            break;
        case 'ArrowRight':
            console.log("RIGHT!");
            break;
        default:
            console.log("IGNORED!")
    }
```

```
input.addEventListener('keydown', function (e) {
    console.log("KEYDOWN")
```

```
KeyboardEvent {isTrusted: true, key: "q", code:
"KeyQ", location: 0, ctrlKey: false, …}
    key: "q"
    charCode: 0
    code: "KeyQ"
```

a	Key
KeyA	Code
	Key
Space	code

Shift	Key
ShiftLeft	Code

Enter ur name Description syed Enter

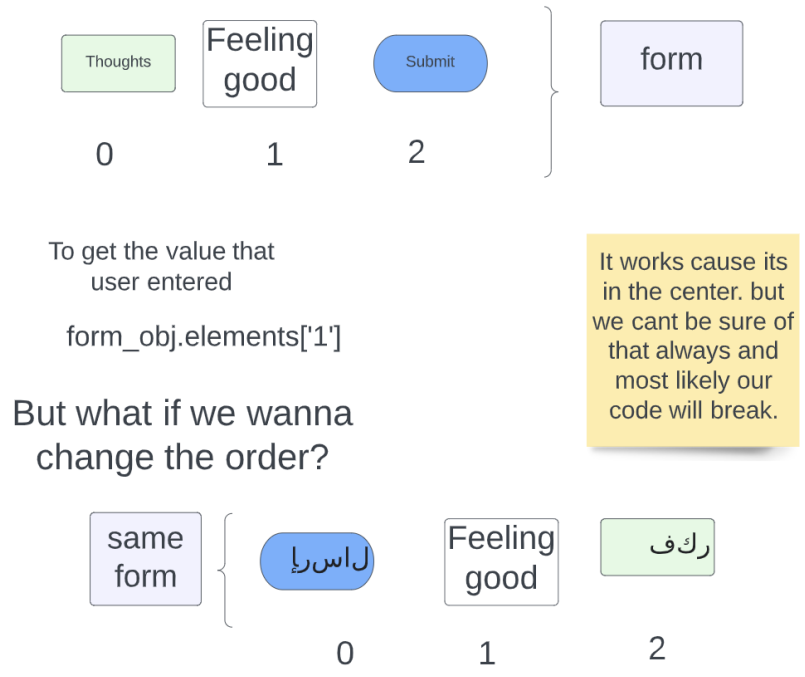
Attributes: {0:action,1:id}
Autocomplete:”on”
Autofocus:false
ChildNodes:Nodelist(11)
Children:HTMLCollection(5)
childrenElementCount

contentEditable:inherit
Draggable:false
Name:””
nextSibling
nextElementSibling
nodeValue
previousSibling
Parent node:SECTION
Parent Element:sECTION

firstChild:text
firstElementChild:label
STYLE
lastChild:text
lastElementChild
Elements
0:input id input_1
1:input id input _dEsc
2: button
tagName:Form

So u have to refer through indexing like form_obj.elements[0].
SO what if, u change the order. The code will break.

Tip:



if u put the name attributel= like form > input[type='text' name="txt_thoughts"] you can access this variable using **form.elements.text_thoughts.value** just like an object's field.

we pass the value from the form in the form of an object.
?name=kazim&age=22.

```
<form action="/dogs">
    <input type="text" name="username" placeholder="username">
    <input type="text" name="tweet" placeholder="tweet">
    <button>Post</button>
</form>
```

```
const tweetForm = document.querySelector('#tweetForm')
tweetForm.addEventListener('submit', function (e) {
    console.log("SUBMIT!!")
    e.preventDefault();
});
```

if action isn't specified it will send data to the page you are on rightnow.

In below, put e.preventDefault() afterbegin inside eventlistener.

```
const tweetForm = document.querySelector('#tweetForm');
tweetForm.addEventListener('submit', function (e) {
    // const usernameInput = document.querySelectorAll('input')[0];
    // const tweetInput = document.querySelectorAll('input')[1];
    const userbane = tweetForm.elements.username.value;
    const tweet = tweetForm.elements.tweet.value;

    // console.log("SUBMIT!!")
    e.preventDefault();
});
```

Tip: The form is submitted by pressing 'Enter' not just by clicking the Submit button.

Input & Change events

There is copy/paste using mouse Or voice over utility, so there are different ways of updating a input and keydown/keyup doesn't encompass all.

EVENT

change:-only fires when you go in and out of input element ***that is*** only if the input was different before you enter the input.
input:-everytime there is a change not just when you go in or out of an input. For e.g: shift or arrow keys wont fire this event. If you cut and paste that will fire this event.

lets say
input[type=’text’ autofocus=true]
so when the page load this input element will be on focus.
Alright.

Lets say you dont have anything to do with this input.
So u goto the console.
No event is been fired.
But lets say u type ur name and get out of the element. You havent pressed enter. It will still fire.
Cause there is a change.
that is only if the input was different from what it was before you enter the input.

Lets say
input[type=’focus autofocus value=’kazim’]
lets you backspace d and then type d without leaving the box focus mode.
So its still kazim
and u went out of it.
The input will not fire.

So if u want the event to be fired
if there is a change and even if u didnt left the focus mode.
Use input
** a live preview.

when if u want the event to be fired
when you get out of the focus mode and it checks whether there is a change.
If it is it will fire, then use change mode.
** wont be able to achieve live preview cuz ill have to leave the tab.

--
you have to use this. Cause keydown won’t work with copy and pasting using mouse.
Keydown and keyup will trigger for every key that is pressed.

Event bubbling

```
<body>
    <section onclick="alert('section clicked')">
        <p onclick="alert('paragraph clicked')">
            I am a paragraph:
            <button onclick="alert('button clicked')">Click</button>
        </p>
    </section>
```

Above, button triggered > para triggered > section triggered.
-Use e.stopPropogation();

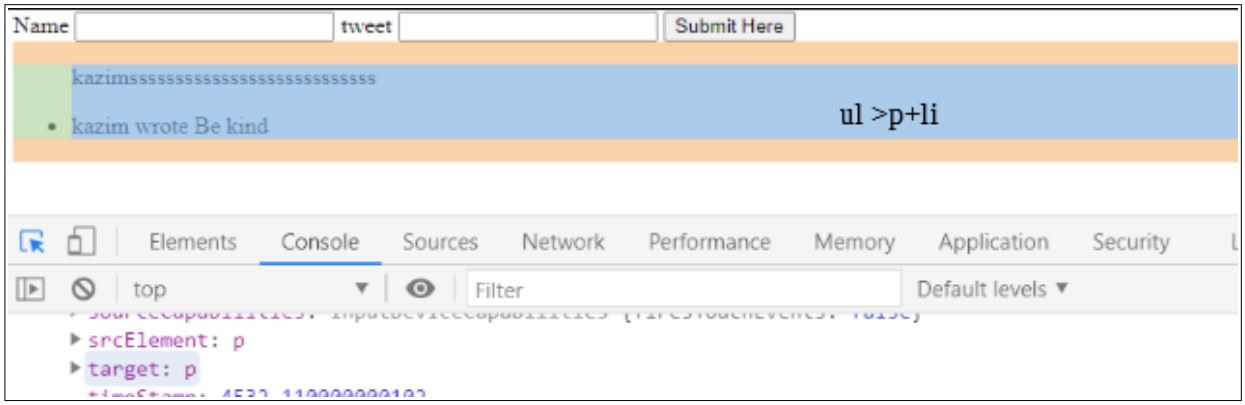
--
Another e.g

div > btn.onclick(giveRandomColor)

div.addEventlistener(‘click’,(e)=>{
div.classlist.toggle(‘hiding’)
}

So if i click on btn nested in the div, the div will dissapear. This is event bubbling.

Event Delegation



Above, on li we have listener which on click are removed. But above if we submit and create a new li, it will not have a listener. So here, we placing a listener on ul.

It means we gonna add event listener to some element that is a parent.
Here, We gonna listen click on ul but specifically on li that is in ul.
So even though i have eventListener on ul, but the target is p here.

```
myul.addEventListener("click",(e)=>{
    e.target.nodeName==='LI' && e.target.remove();
});
```

Working with select

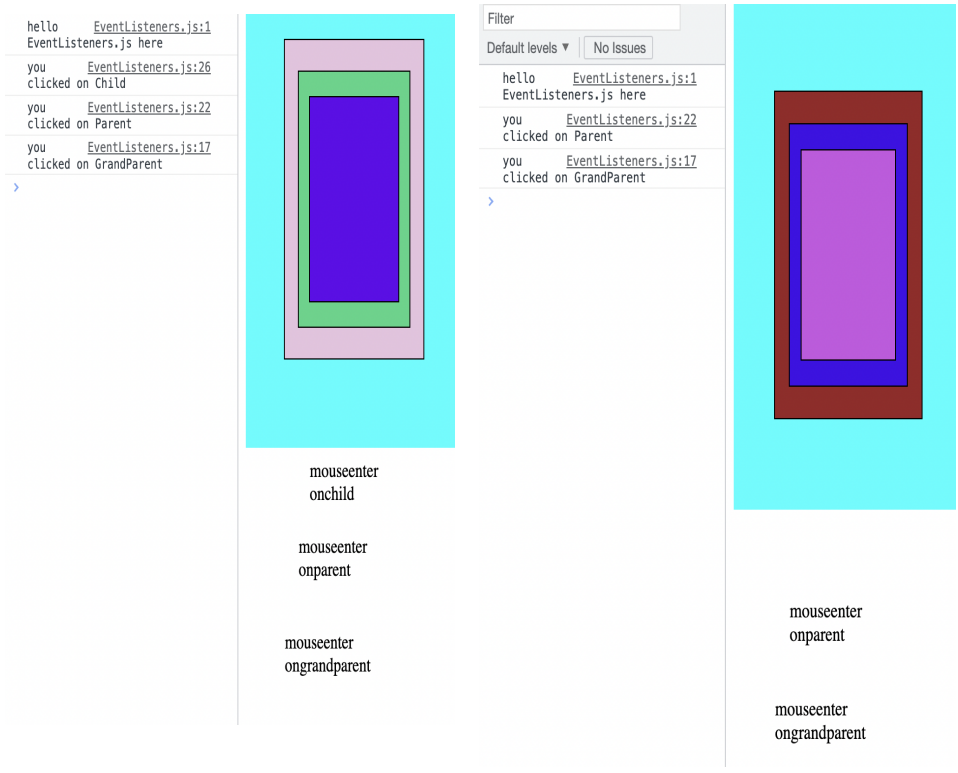
```
myselect.addEventListener("change",function(){
    // console.dir(e.target.options.selectedIndex);
//    max=e.target[e.target.options.selectedIndex].value;
    max=this.value;
    console.dir(this);
});
```

AddEventListener options:

The "capture" option in `addEventListener` is a boolean value that specifies the phase of event propagation to listen for the event.

- If the capture value is set to `false` (default), the event listener will be triggered during the bubbling phase, i.e., it will be called when the event has propagated from the target to the top of the DOM tree.
- If the capture value is set to `true`, the event listener will be triggered during the capturing phase, i.e., it will be called when the event is first captured and travels down the DOM tree towards the target.

It allows you to control the order in which event handlers are called, as well as the direction of event flow (from the top of the DOM tree down, or from the target up)



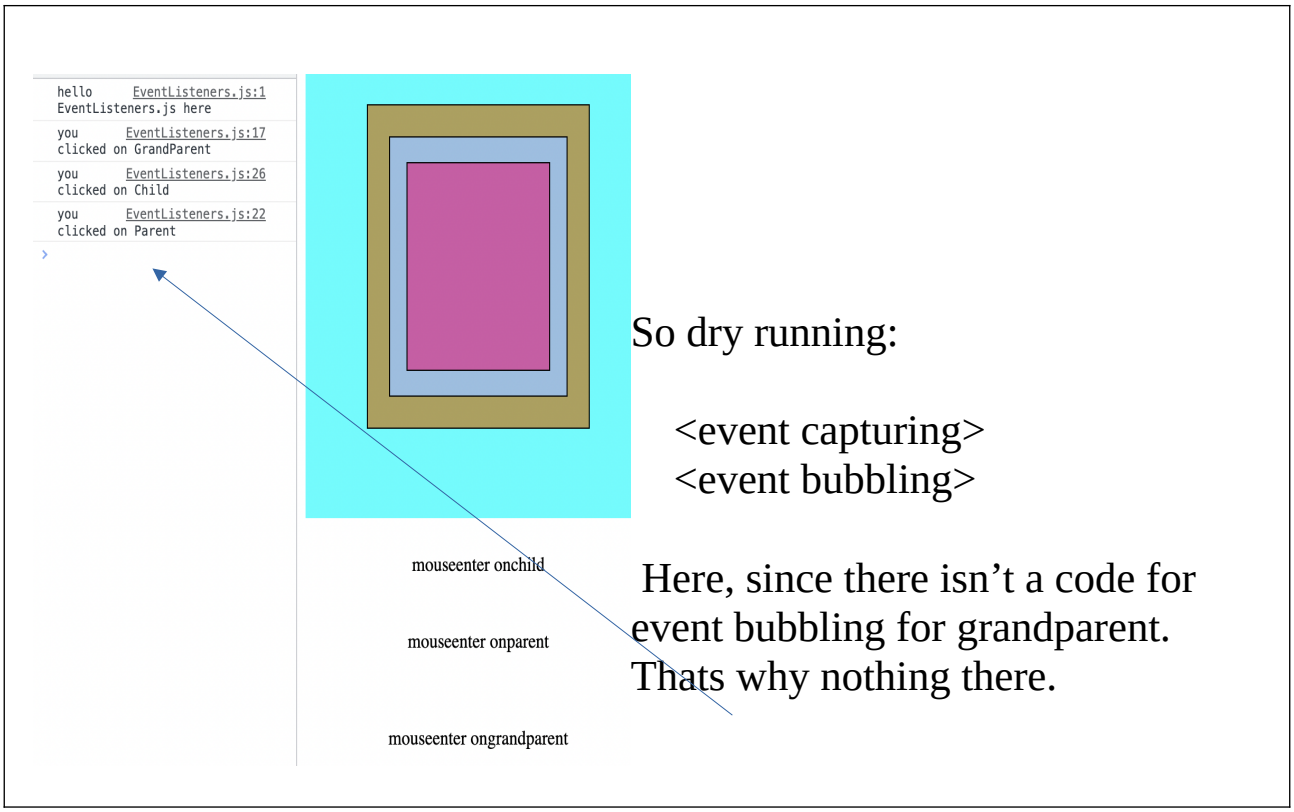

```
x[0].addEventListener('click',(e)=>{
    console.log("you clicked on GrandParent");
});


x[1].addEventListener('click',(e)=>{
    console.log("you clicked on Parent");
});

x[2].addEventListener('click',(e)=>{
    console.log("you clicked on Child");
});
```

This is event bubbling.

Event Capturing using the capture: true option in addEventListener:

The event triggered propogates from top of the dom tree.
It doesnt mean event bubbling is ignored. They both happened.

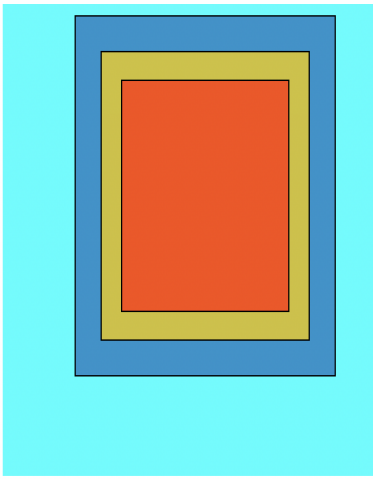
More likely it isn’t handled cuz the default value for capture: false



Code:

```
x[0].addEventListener('click',(e)=>{
    console.log("you clicked on GrandParent");
},{capture: true});


x[1].addEventListener('click',(e)=>{
    console.log("you clicked on Parent");
});

x[2].addEventListener('click',(e)=>{
    console.log("you clicked on Child");
});
```

```
undefined
you        EventListeners.js:17
clicked on GrandParent with
event capturing
you        EventListeners.js:31
clicked on Child
you        EventListeners.js:27
clicked on Parent
you        EventListeners.js:22
clicked on GrandParent with
event bubbling
```



mouseenter onchild

mouseenter onparent

mouseenter ongrandparent

code:

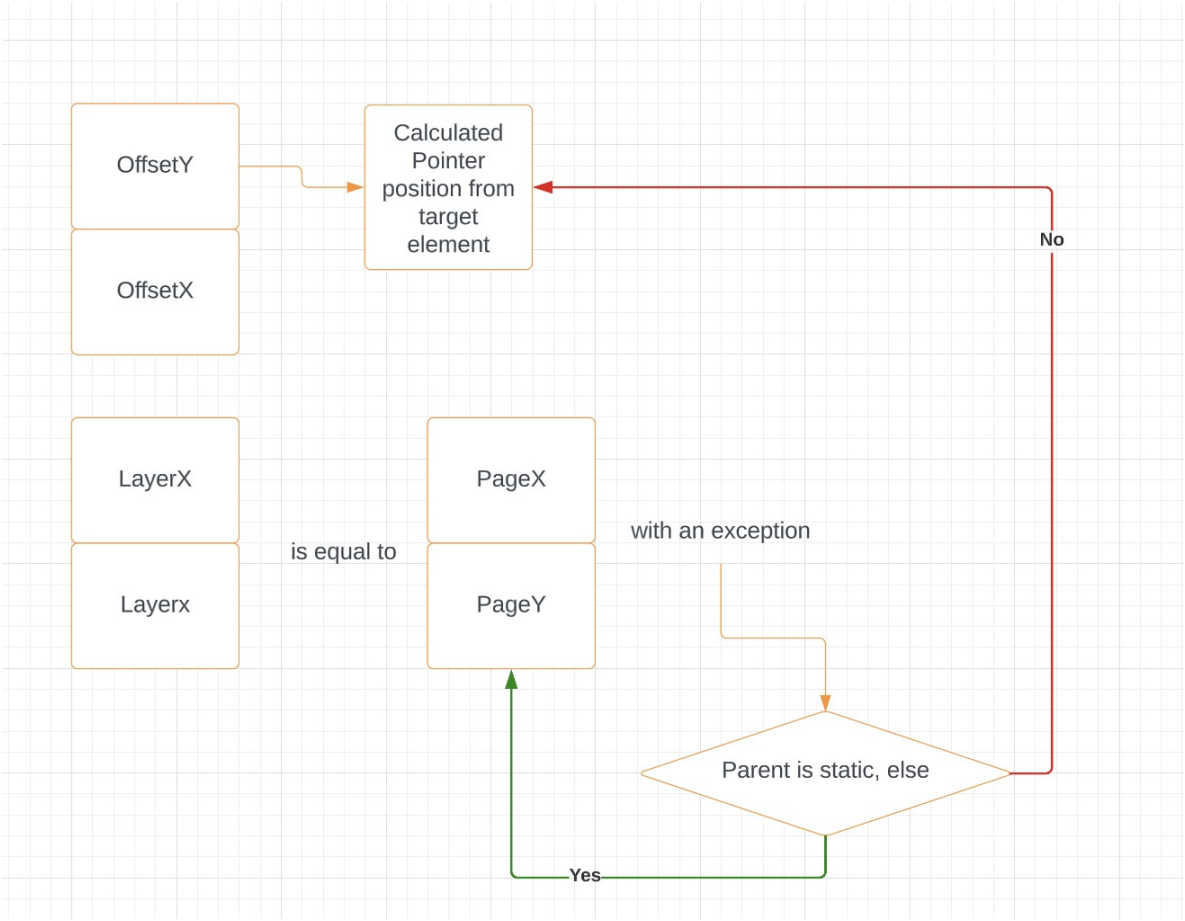
```
x[0].addEventListener('click',(e)=>{
    console.log("you clicked on GrandParent with event capturing");
},{capture: true});


x[0].addEventListener('click',(e)=>{
    console.log("you clicked on GrandParent with event bubbling");
},{capture: false});
```

for a listener to be called only once.

```
x[0].addEventListener('click',(e)=>{
    console.log("you clicked on GrandParent with event bubbling");
},{capture: false,once:true});
```

PointerEvents:

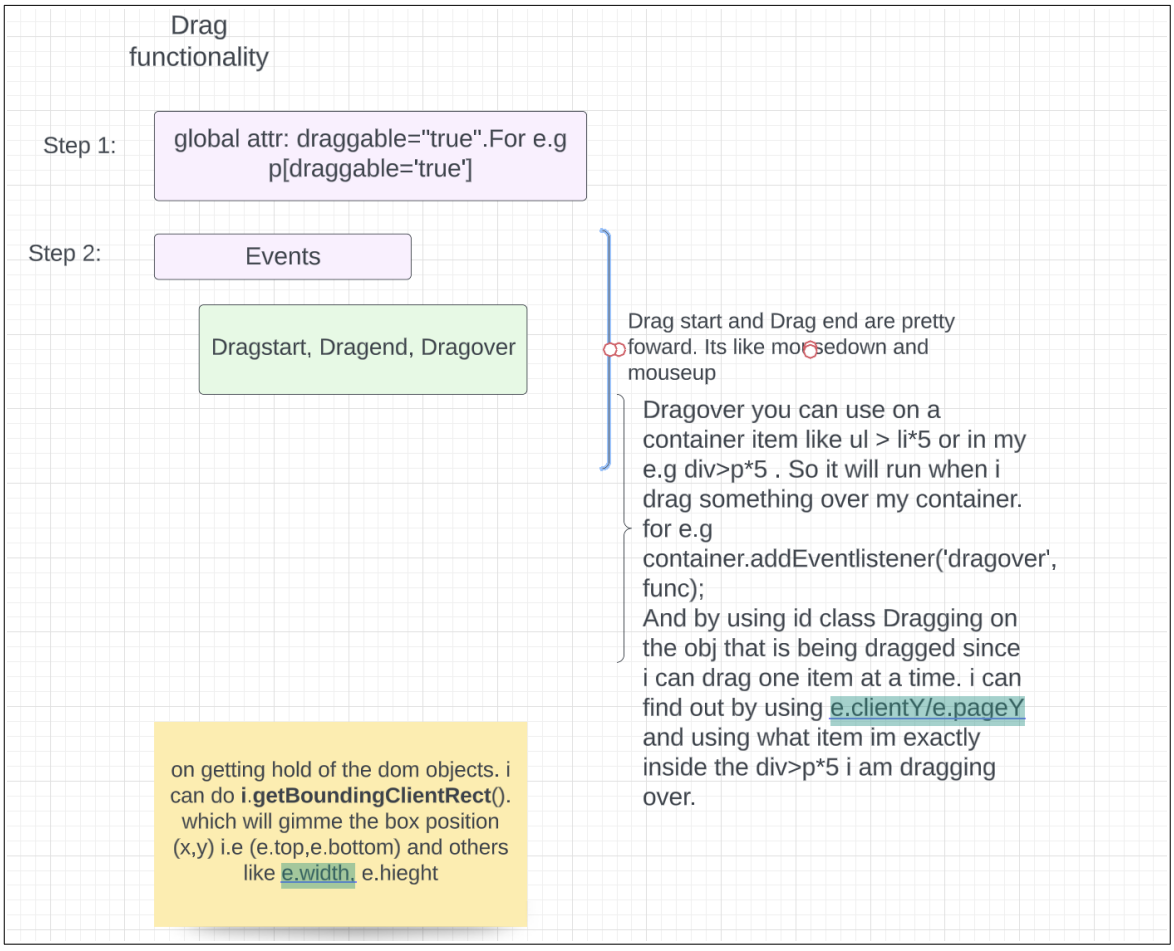
LayerX and LayerY Retrieves the x-coordinate, y-coordinate respectively of the mouse pointer relative to the top-left corner of the closest positioned ancestor element of the element that fires the event.



To use the drag functionality.

```
el=document.getElementsByTagName('h1')
▶HTMLCollection [h1]
el[0].style.border='1px solid black';
'1px solid black'
el[0].getBoundingClientRect().top
99.375
document.addEventListener('click',(e)=>{
    console.log('i am being clicked',e.pageY)
});
undefined
i am being clicked 101                                VM1554:2
```

Figure 2: I clicked right on the border above H



innerHTML vs innertext vs textContent