Javascript

→ document.getElementbyId(" "). =

Ways:

① → <script> ... </script> in HTML body

② → index.js file

<script src = "index.js"> </script>

→ Comment : //

→ Variables:

let myAge = 20
console.log(myAge)

✱ Click event

html: <button onClick = "increment()">

js:

function increment() {

    console.log("Button clicked")

}

✱ Increment button JS

let countEl = document.getElementById
("count-el")

in HTML

```
let count = 0

function increment() {
    count += 1
    countEl.innerText = count
}
```

countEl.innerText → count 1, 2, 3 ...

→ HTML:

```
<h2 id="count-el">0</h2>
<button onclick="increment()">INCREMENT</button>
<script src="index.js"></script>
```

✱ Document Object Model (DOM)

→ How you use JS to modify a website.

Document → HTML Document

Object → document = { } → object

(Document keyword in javascript is of datatype object.)

Model → Representation

Real:

```
<h2 id="count-el">0</h2>
```

Model:

```
let countEl = document.getElementById("count-el")
```

CamelCase

Page No.:

✱ String and numbers :

~~let points = 4~~
~~let totalPoints =~~

let points = 4

let bonusPoints = "10"

let totalPoints = points + bonusPoints
(4) ("10")

console.log (totalPoints)

⟶ 410

Between no and string, string always wins.
i.e. points is converted into string "4".
∴ "4" + "10"
= 410

✱ Variables and functions are written in camelCase

~~e.g.~~ first letter small, second Uppercase

e.g. totalPoints

fullPrice

bonusPoints

Page No.:

✱ Difference btn Innertext and textContent

→ textContent : Contents all the elements, including <script> and <style> elements
↳ more preferable

→ innerText: Only shows 'human-readable' contents. Won't return text of 'hidden' elements like spaces

✱ If-else statements

```
if( age < 100){

}else if(      ){

} else {

}
```

age === 100
↳ always use this (strictly equal)
Good method

age == "100" ✕ → This will return true i.e. age is equal to 100 though it is string
age === "100" ✓ This will return false
It won't consider string as a number 100 i.e. strictly equal

✱ Ways to select / grab elements from DOM:

(2) let sum =

1) document.getElementById("sum-el")
↑ id in HTML

2) document.querySelector("#sum-el")
↓ id

" . ".sum-el")
↓ class

" . "body")
↓ Whole tag

✱ Arrays

→ Ordered lists of items

e.g. let featuredPosts = [
"Check out Netflix clone",
"Here's the code",
"Portfolio"
]

→ 0 indexed (counting starts from 0)

→ featuredPosts.length ⟹ 3
↳ +1 array index

→ Primitive Data types → string
→ number
→ boolean

→ Array → Complex/composite data type

e.g. let Khushi = ["Khushi", 20, true]
string number boolean

✱ Push:

```
let cards = [7,4]
cards.push(6)
console.log(cards)
```

ArrayName.push(_)
↑ to add items

→ [7,4,6]

cards.push(6)
↑ object ↑ method

✱ Pop

cards.pop() → Removes the last item of the array

✱ FOR loops:

start finish step size

```
for(let count = 1; count < 11; count += 1){
    console.log(count)
}
```

i=1 i<11 i+=1 or i++

* Generate random numbers :

Math.random()

let randomNumber = Math.random()

console.log(randomNumber)

→ Always gives ans between 0 to 1 excluding 1.

0.000 ⟶ 0.999

Math.random() * 6 ⟹ 0.000 → 5.999

⟶ Math.floor(9.6369) = 9
" " (12.5675...) = 12

∴ Math.floor() removes the decimal values.

⟶ Math.floor(Math.random() * 6)

⇩

0 | 1 | 2 | 3 | 4 | 5

⟶ How to get 1 to 6 (for dice)

Math.floor(Math.random() * 6) + 1

* Logical AND — option (not needed)

```
if( ____ === true && ____ === true){
    ____
}
```

* Logical OR (||)
* Objects → Complex / composite data type

```
let player = {
    playerName : "Per",
    chips : 145
}
```

(chips: Key, 145: value)

```
playerEl.textContent = player.name + ": $" +
                       player.chips
```

→ Key - value pairs

→ console.log(player.name)

→ Function in object :

```
let player = {
    sayHello : function() {
        console.log("Heisann")
    }
}
```

Object function

player.sayHello()

Soln: ⟹ Heisann

console.log()
object function

✱ Shift, unshift in array

(1) shift() → Removes an item from the beginning of the array

(2) unshift( ) → Adds the element at the beginning of the array (moves the 1st elt into 2nd place and so on)
↙ what to add

(3) pop() → Removes last element of the array

(4) push( ) → Adds new element at the end of the array.
↑ the element to add

✱ AddEventListener:

```
let inputbtn = doc.getEleById("inp-btn")
inputbtn.addEventListener("click", () => {
    console.log("Button clicked")
})
```

"click", () => { } → Can also be written as function()

"click" → event, function

✡ const and let

→ const cannot be reassigned (its constant)

→ let can be reassigned

→ If possible, use const. If not, use let.

✡ Get value from input field.

html:

```
<input type="text" id="input-el" >
```

js:

```
const inputEl = doc.getEleById("input-el").value

inputBtn.addEventListener("click", function(){ option

    myLeads.push(inputEl.value)

}
```

Simple: .value

```
doc.getEleById("id-name").value
```

* innerHTML

____.innerHTML = "<button>" + ____ + "</button>"

↓

We can add into HTML page (tags)

____.innerHTML = "<button>Buy!</button>"

↓

Will create a [Buy!] button

* Use createElement() and append() instead of .innerHTML

→ ① create element
② Set text content
③ Append to the ul

① document.createElement("li")

~~②~~ const li = document.createElement("li")

② li.textContent = myLeads[i]

③ ulEl.append(li)

→ .innerHTML easy & simple to use

✱ Template Strings ` ` / Template Literals

e.g. Normal:

```
<"li><a target='_blank' href='" + myLeads[i]
    + "'>" + myLeads[i] + "</a></li>"
```

Problem: Single line
Complex
Hard to understand

Using template strings:

```
`<li>
    <a target='_blank' href='${myLeads[i]}'>
        ${myLeads[i]}
    </a>
</li>`
```

Advantages: Multiple line allow
Simple & easy
No need to worry about ', ", +

Instead of + myLeads[i], here
${myLeads[i]} is use

escapes the HTML and
sees in JS.

✱ JSON

→ JavaScript Object Notation

→ Way for developers to store and send data

→ Often use when we want to send a piece of data from a server to a client.

✱ Local Storage

→ LocalStorage is a data storage type of web storage.

→ This allows the JS apps and sites to store and access the data without any expiration date.

→ Thus, data stored in the browser will be available even after closing the browser window.

→ How to write in JS:

1) setItem (localStorage.setItem(key, value)

localStorage.setItem("myname", "Khushi")

2) getItem (LoSto.getItem(key))

let name = localStorage.getItem("myName")
console.log(name)

3) clearItem (toStr.clear())

localStorage.clear()

* Both key and value needs to be strings * (in double quotes)

* JSON.parse() ⇒ Convert a string into Javascript object

JSON.stringify() ⇒ Convert a javascript object into a string

⇒ The keys and values stored with localStorage are always in the UTF-16 string format.

* How to find type of a variable in JS?

console.log(typeof ____)

* How to store array in local storage.

localStorage.setItem("myLeads", JSON.stringify(myLeads))
Key array

Optional: (See if its working)

∴ console.log(localStorage.getItem("myLeads"))
Key

✱ Truthy and Falsy values

→ Falsy values:

- false
- "0"
- null
- undefined
- NaN (not a number)

→ Most used

→ null: how you as a developer signalize emptiness

→ undefined: how javascript signalizes emptiness

e.g. let viewers = null
viewers = ["Khushi, "Tanvi"]

truthy value (array)
if(viewers) {
    console.log("We have viewers")
}

Falsy (null) — If we set it to null, then this if won't run
viewers = null

e.g. let viewers → not defined
console.log(viewers)
→ undefined

→ How to check Truthy or Falsy ?

let trueOrFalse = Boolean ( ) ← variable, value

"", [], null

console.log(trueOrFalse)

Boolean (-0) ⇒ false

✱ Double click event listener (dblclick)

_____.addEventListener("dblclick", () ⇒ { })

✱ Function Parameters

e.g function greetUser(greeting, name) {

welcome.textContent = `${greeting}, ${name}!`

}

greetUser("Hello", 'Khushi")

Output :

Hello Khushi!

✱ Arguments vs Parameters

→ Parameters : Names listed in the function's definition

parameters

function greetUser (greeting, name)

→ Arguments: Real values passed to the function.

Outside of the function

e.g. greetUser("Hello", "Khushi")
arguments

⁂ How to get current tab link in our extension / application using JS?

chrome.tabs.query({active: true, currentWind
obj obj method

: true}, function(tabs){

tabs[0].url

})

→ manifest.json ⇒ Extension mutadata is defined here

Add:

"permissions": [

"tabs"

]

}

Round Number in JavaScript

1. Math.round(number) e.g. 2.356 = 2
2. Upto 2 decimal places

Math.round(number * 100) / 100

e.g. 2.456789 = 2.45

* Different method: (number).toFixed( )
  ↓ decimal place

e.g. Upto 2

2.456789 = '2.45'

⇒ Math.round ⇒ Works both on number and string
i.e. num = 2, num = "2"

toFixed ⇒ Only on number.

If string given then, e.g num = "2.456"

Number(num).toFixed(2)

convert string to number