

Javascript

Page No.:

→ 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.innerHTML = count + 1

}

→ HTML: $\text{Count} + \text{string} = \text{CountEl.innerHTML}$

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

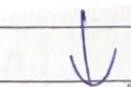
<button onclick="increment()">INCREMENT</button>

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

* Document Object Model (DOM)

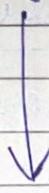
→ How you use JS to modify a website.

Document Object Model



HTML

Document



Representation

document = { } ↓

Real:

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

{ Document keyword
in javascript is
of datatype
object. } ↓

object

Model:

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

Camel Case

Page No.:

* String and numbers : $0 = \text{four} + 1$

let points = 4 } () + number addition
let totalPoints = }
 + - + four }

let points = 4 * 1.00 + 1 * 1.00

let bonusPoints = "10"

let totalPoints = points + bonusPoints

console.log (totalPoints)

$\Rightarrow 4 + 10 (" + number)$ Between no and string, string always wins

$\Rightarrow 4 + 10 (" + string)$ i.e. points is converted

into string "4" + "10"
 $\therefore "4" + "10"$

* Variables and functions are written in camelCase

first letter small, second uppercase

e.g. totalPoints

fullPrice

bonusPoints

* Difference b/w Inertext and textContent

→ `textContent`: Contains 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 = document.querySelector('#sum-el') id in HTML

i) document.getElementById("Sum-el")

ii) document.querySelector("#sum-el")

 id

 class ".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 or 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

→ [7, 4, 6]

cards.push(6)

↑
object method

* Pop

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

* For Loops:

start with i=1, finish i<11, step size i+=1
for (let count = 1; count < 11; count += 1) {
 console.log(count)
}

or
i++

* Generate random numbers:

Math.random()

let randomNumber = Math.random()

console.log(randomNumber)

→ Always gives ans between 0 to 1

excluding 1

→ New code point $[0, 1) = \{0.000, 1.000\}$

$0.000 \rightarrow 0.999$ (Ans) $\{0.000, 1.000\}$

But Math.random() * 6 $\Rightarrow 0.000 \rightarrow 5.999$

→ Math.floor(9.6369) = 9

" " (12.5678...) = 12.678

∴ 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 ~~returning~~ option (not needed)

if (~~if (~~ == true: ~~if (~~ == true) {

}

para si Hidam, Hidam

* Logical OR (II) variale - Hidam (I)

* Objects → Complex / composite data type

let player = {

playerName: "Per",

chips: 14500

} Key value

player.E1.textContent = player.name + ": \$" +
player.chips

→ Key-value pairs

→ console.log(player.name);

→ Functions in object : sayHello();

let player = {

sayHello: function() {

console.log("Neisann")

}

}

object function
player.sayHello()

SOLN: \Rightarrow Neisan \rightarrow console.log()
object function

* Shift, unshift in array

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

② unshift() \rightarrow Adds the element at the beginning of the array (moves the 1st element to 2nd place and so on)

③ pop() \rightarrow Removes last element of the array

④ push() \rightarrow Adds new element at the end of the array

* AddEvent Listener

```
let inputbtn = doc.getEleById("inp-btn")
```

```
inputbtn.addEventListener("click", () => {
```

```
    console.log("Button clicked")
```

}

"click", () => {}

event function

Can also be written as
"click" function if

* 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. using.

html: id [type] is stored like

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

js:

const inputEl = doc.getElementById("input-el")

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

("id") myleads.push(inputEl.value); })

}

[i] about you = dragging + text : ()

Simple: .value

doc.getElementById("id-name").value

* innerHTML

(innerHTML).innerHTML = "<button>" + text + "</button>"

↳ dangerous as it

We can add into HTML page (tags)

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

Will create a Buy! button

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

→ ① createElement

② Set text content

③ Append to the ul

① document.createElement("li")

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

③ li.textContent = myLeads[i]

④ ulEl.append(li)

→ .innerHTML easy & simple to use

* Template Strings / Template Literals

e.g. Normal:

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

Problem: Single line

Complex

Hard to understand

Using template strings:

```
'|i>
```

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

```
|/a>
```

```
|/i>
```

Advantages: Multiple line allowed

Simple & easy to read

No need to worry about ', ", +

Instead of + myLeads[i], here

`\${myLeads[i]}` is used

escapes the HTML and

uses in JS

* JSON

→ JavaScript Object Notation

→ Way for developers to store and send data

→ Often use when we want to send or 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", "Khusi")

2) getItem (localStorage.getItem(key))

~~Page No.:~~

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

3) clearItem (localStorage.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 variable)

* How to store array in Local storage.

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

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 // viewer bro niet erst

viewers = ["Khuski", "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 checkTruthy or Falsey ?

let trueOrFalse = Boolean ()
Assignment with the 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

function greetUser (greeting, name) {
parameters

→ 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, currentWindow: true}, [obj] method
  [obj, printing: true], function(tabs) {
    tabs[0].url
  }
}
```

→ manifest.json ⇒ Extension metadata 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()`
 \downarrow
decimal place

e.g. Upto 2

$2.456789 = '2.45'$

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

`toFixed` \Rightarrow Only on number.

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

`Number(num).toFixed(2)`

convert String to number