Javascript

- Introduction to Javascript :→

1. Js in console
2. Dom manipulation
3. chrome extensions
4. what is a programming language?
5. what is syntax?
6. HTML/CSS/Javascript

1. Js in console

* Ex: alert ("Hello Javascript");

- → Console can be used as a calculator :→

Ex.: > 2 x 3
• 6

2. Dom manipulation

↓

Document object model

Ex: document.body.innerHTML = 'Welcome to KG coding'
<· 'welcome to KG coding'

document.body.innerHTML = ' Welcome to KG coding '
<· ' welcome to KG coding '

3. Chrome extensions

* Create features: Add new functionalities to chrome.
* Interact with web: modify or read webpage content.
* API Access :- use chrome's built-in function
* User Experience :-

4. What is a programming language?

→ Giving instructions to a computer

→ Instructions: Tells computer what to do.

→ These instructions are called code.

5. what is a syntax?

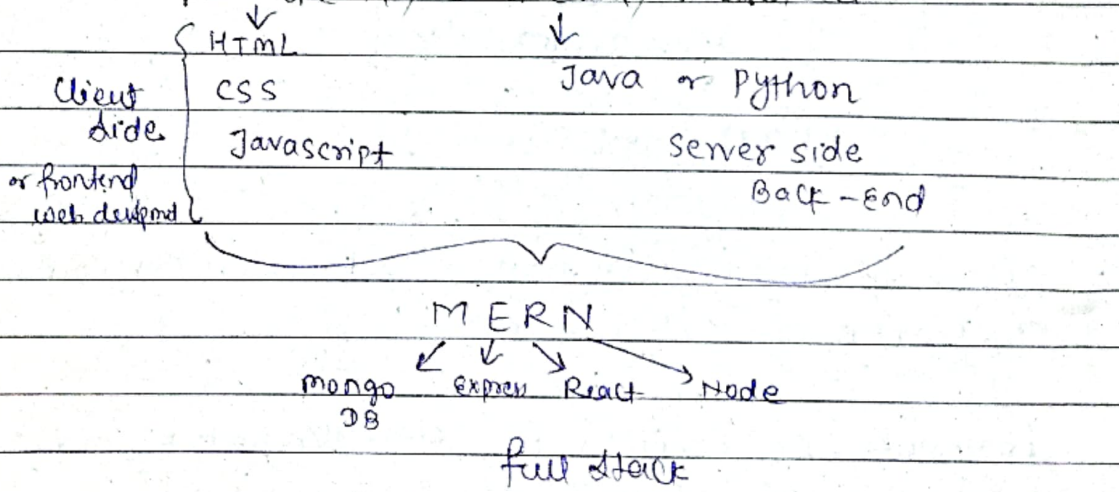
→ structure of words in a sentence.

→ Rules of the language.

→ for programming exact syntax must be followed.

Ex: alert "hello world".

6:→ frontend / Backend / fullstack :→



6:→ Role of Browser :→

HTML - CSS - Javascript → decode → Browser → Rendering → web page

- Displays web page: Turns HTML code into what you see on screen.
- user clicks: Helps you interact with the web page
- updates content: - Allows changes to the page using Javascript.
- Loads files :→ Gets HTML, images, etc., from the server.

→ Java script :→

1. Javascript has nothing to do with Java.
2. Actions: enables interactivity.
3. updates: Alters page without reloading.
4. events: Responds to use actions.
5. Data: fetches & sends info to server.

code → compile — executable
code → transpilex → code

→ Javascript runs at client side in the browser.
→ coffeescript / Typescript are transpiled to Javascript.

1. Practice exercise

- use an alert to display Good morning.
- Display yourname in a popup.
- perform maths function
  ⇒ 75 − 25 , 3 + 3 − 5
- Change facebook page to display "I am learning JS".

Numbers & strings

- Arithmetic operations
- order of operations
- Types of Numbers (Integers & floats)
- Don't learn syntax.
- Strings
- Type of operators.

* Arithmetic operators :→

"+" → Addition → 4+2 → 6

"−" → substraction → 4−2 → 2

"*" → multiplication → 4*2 → 8

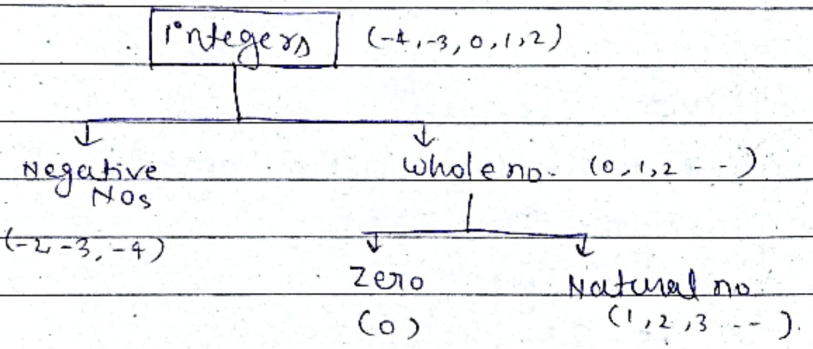
"/" → Division → 4/2 → 2

"%" → modules → 5%2 → 2

* order of operation :-

B	O	D	M	A S
↓	↓	↓	↓	↓ ↓
( )	$\sqrt{x}$ x^2	÷ or *	mult	Add Subs

* Types of Numbers :→



Ex.: In google Console :→

> Math.round(180.32)

< 180

* Dont learn the syntax :→

- Google :-
- MDN → In depth (read the documentation)
  https://developer.mozilla.org
- ChatGPT : Real time amistance for coding queries.
- focus : understand concepts, not just syntax.

11 Strings :→

Ex.: | E | X | A | m | P | L | E |

- String hold textual data, anything from a single character to paragraph.
- String can be defined using single quotes '', double quotes "", or backticks ``.

Ex:- "Hello world"
'Hello world'
`Hello world`
"You're welcome"

Ex-: '₹' + (5.9 + 0.1)	` ₹ ${5.9+0.1}`
⇒ '₹6'	⇒ '₹6'

Ex: 'Goodmorning' + 'Sanchit sir'
<- Goodmorning sanchit Sir

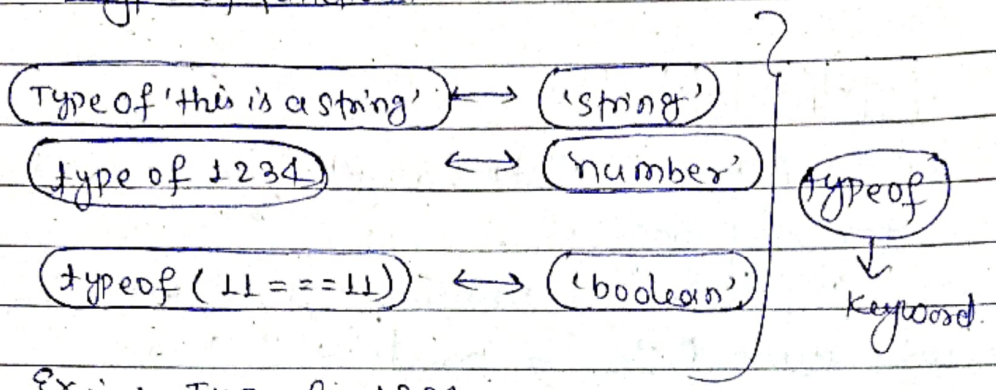
Ex: > 4+9	> '4' + '9'	> 'You have got' +4.5+ 'Stars out of 5'
<- 13	<- '49'	<- You have got 4.5 stars out of 5

Ex: > 'you have got ' + 1 + 1 + 1 + 'votes'
<- You have got 111 votes
Ex: > 'you have got ' + (1+1+1) + 'votes'
<- 'You have got 3 votes'

12 Types of operator :→

1. Check Type : Tells you the data type of a variable
2. Syntax : use it like typeof variable
3. Common Types : Returns "number," "string," "boolean," etc.

JS | Types of function



Ex.:- > Typeof 1234
<- 'number'

* * * * JavaScript with HTML & CSS * * * *

- Vs Code IDE (Intigrated development environment)
- HTML Tags Introduction
- CSS Introduction
- Script Tag
- Comments.

* IDE :-> Intigrated development environment
* Central hub for coding, finding problems, & testing.
* Designed to improve developer efficiency.

* Code automation
* Syntax Highlighting
* Version control
* Error checking.

* HTML DOM :-

- Structure understanding : Helps in understanding the hierarchical structure of a webpage, crucial for applying targated css styles. & dynamic style, means real time changing & interactivity through css.

```
h1 {
  font-size : 30 ;
}
```

selector — declaration (property : value)

16. Query Selector Javascript

1. getElementById : finds one element by its ID.
2. getElementByClassName : finds elements by their class. returns a list.
3. queryselector : finds the first element that mathes a CSS selector.
4. Purpose : To interact with or modify webpage elemt

```
Ex-: >document.getElementById('click-me')
<.  <button id = "click-me"> click me </button>
>   document.getElementById('click-me').click()
```

```
Ex: >document.querySelector('h2')
<. - - -
>document.querySelector('#click-me')
<.
>document.querySelector('.click-me')
<.
> document.querySelectorAll('.push-me')
```

17 Script Tag

1. Embed Code : Incorporates javascript into an HTML file, either directly or via external files.
2. Placemet : commenity placed in the <head> or just before the closing </body> tag to control when the script runs.

3. External files : use src attribute to link external Javascript files, like <script src = "script.src = "script.js"></script>

4. Console methods: log, warn, error, clear

Ex: console.log('Hello Friends');
console.log (2+2);
Console.log ('Good Morning' + 'Priyanshu');
console.error('this is error');
Console.warn('warning');

** Comment in JS/CSS :—
1. // Console.log('Hello');
2. /* alert('Priyanshuraj') */

** in HTML: →
<!-- hello -->

*** Variables ***

- What are variables
- Syntax Rules
- updating values
- Myntra Bag exercise
- Naming conventions
- ways to create variables.

→ variable is used to store Data.

Variable

Ex:→
```
<Script>
let bagsalary = 'Total amount is ${2+2/3}';
Console.log (bagsalary);
</Script>
```

1. Can't use keywords or reserved words.
2. Can't start with a number.
3. No special characters other than $ and _ = is for assignment
4. ; means end of instruction

Ex:
```
let a =3;
let b =3;
console.log (a+b);
```

* Practice exercise :-
→ Myntra Bag

23 Naming Conventions :→

① Camelcase:→
- start with a lowercase letter. Capitalize the first letter of each subsequent word.
- Ex: myVariableName

② Snake_case :→
- Start with an lowercase letter. separate words with underscore
- Ex: my_variable_name

③ Kebab-case :→
- All lowercase letters. separate words with hyphens. use for HTML and CSS
- Ex: my-variable-name.

④ Keep a Good and short Name :→

- choose names that are descriptive but not too long.
  It should make it easy to understand the variables purpose.

ex:- age, firstName, isMarried

24 Ways to create Variables :→

ex:-
```
<script>
    let P = 3.14;
    let r = 4;
    console.log(P * r * r);
</script>
```

ex:-

```
let radius = 4;
```

```
Const Pi = 3.14
```

```
var
```

ex:-

To print on console

```
let myName = 'priyanshu';
Console.log(myName);
```

To Print on webpage

```
let myName = 'Priyanshu';
document.querySelector('#name-display')
    .innerText = myName;
```

25 Project (Calculator) :→

26 If - else & Boolean :-

→ Data Type : Booleans are a basic data-type in Javascript. (true / false)

→ 'true' is a string not a boolean.

ex:- console.log(5>9); →& false.

ex:- console.log(typeof 'true');

26 Comparision operations :-

<, >, <= , >= , == , !=

- Equality :→

  == checks value equality.

  === checks value & type equality.

- Inequality :→

  != checks value inequality.

  !== checks value & type inequality.

- Relational :→

  > Greater then.

  < less than.

```
console.log(5 == '5.0');
console.log(5 == '5');
← true

console.log(5 === '5.0');
console.log(5 === '5');
← false
```

27 gf-else :→

- Syntax : uses gf () {} to check a condition.

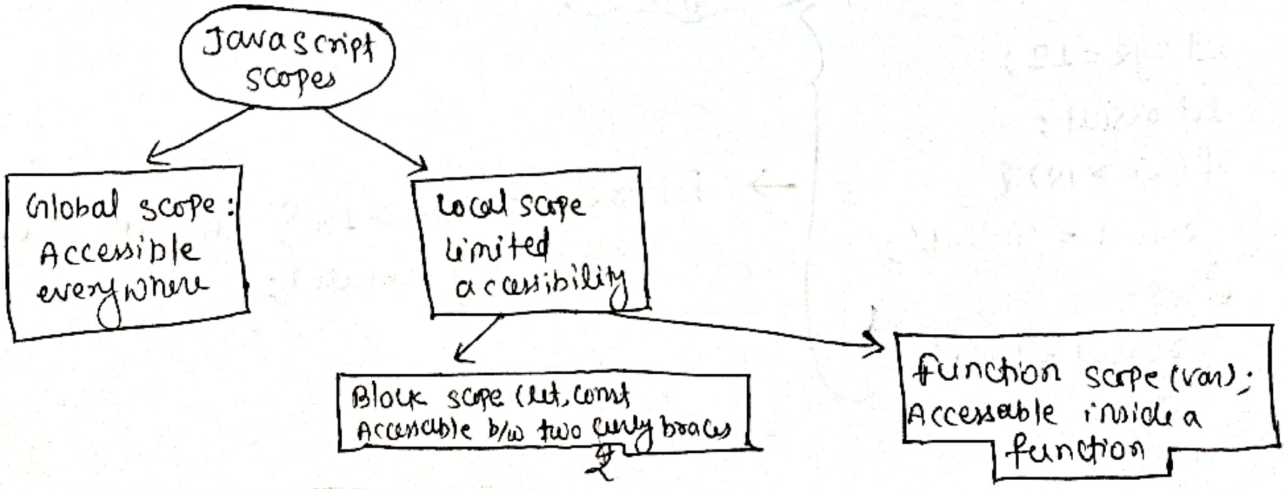
28 Bat-ball-stump project :→

→ math.random() ;

- logical operators :→

① && (AND) , || (OR) , ! (NOT)

29 Scope :→



1. Any variable.
2. Variable can be redefined inside { }
3. Var does not follow scope.
4. Global scope: Accessible everywhere in the code.
5. Block scope: Limited to a block, mainly with let & const.
6. Declare variables in the narrowest scope possible.

30. Truthy & falsy values

1. Falsy values: 0, null, undefined, false, NaN, "" (empty string).
2. Truthy values: All values not listed as falsy.
3. Used in conditional statements like if.
4. Non-boolean values are auto-converted in logical operations.
5. Be explicit in comparisons to avoid unexpected behavior.

31. If alternates

1. Ternary operator: condition ? trueValue : falseValue
   Quick one-line if-else.
2. Guard operator: value || defaultValue
   use when a fallback value is needed.
3. Default operator: value ?? fallbackValue
   use when you want to consider only null and undefined as falsy
4. Simplifies conditional logic.
5. Use wisely to maintain readability.

Ex:-

```
let age = 12;
let result;
if (age > 18) {
  result = 'Adult';
}
else{
  result = 'Kid'
}
```

Ternary operator

→

```
let result = age > 18 ? 'Adult' : 'Kid';
Console.log(result);
```

(ii) Nullish operator :-

```
let age = 0;
let finalAge = age || 18;
console.log(age);
console.log(finalAge);
```

(iii) Default operator :-

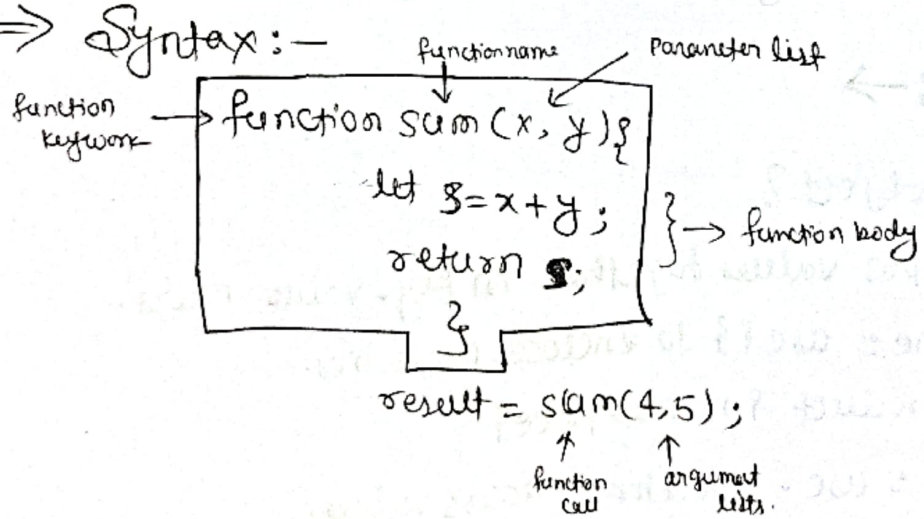
```
let age = 0;
let finalAge = age ?? 18;
console.log(age)
console.log(finalAge);
```

* Functions :→

- What are functions
- function syntax.
- Return statement
- function parameters.

→ • Definition : Blocks of reusable code.
- Dry principle : "Don't Repeat yourself" it encourages code reusability.
- usage : organizes code and performs specific tasks.
- Naming Rules : Same as variable names : camelCase
- example : "Beta chai banana hai".

⇒ Syntax :-



⇒ Parameters :-

- Input values that a function takes.
- Parameters put value into function, while return gets value out.
- Naming convention : same as variable names.

* Practice exercise :-

① create a method to check of a number is odd or even.

② create a method to return large of the two numbers.

③ create method to convert celsius to fahrenheit $f = (9/5)*C+32$.

ex:
```
function large (num1, num2) {
    if (num1 > num2) {
        return num1;
    }
    else {
        return num2;
    }
}
console.large(large(num1, num2));
```

Ex-:
```
function tofahrenheit(C)); {
    return (9/5)*C+32;
}
console.log(tofahrenheit(3C));
```

** Objects :→

[36] What is an object?

→ Groups multiple values together in key-value pairs.
→ How to Define : use {} to enclose properties.
→ Example : product { name, price }
→ Dot Notation : use . operator to access values.
→ Key Benefit : organizes related data under a single name.

object name ← let product = {

object { key : value ← company : 'mango',
item_name : 'cotton striped t-shirt',
price : 861
};

```
let user = {
    name: "John",
    age : 40
};
```

(38) **Accessing objects :-**

→ Dot Notation : Access properties using . operator

like : product.price

→ Break Notation : useful for properties with special characters product ["nick-name"]. variables can be used to access properties.

→ type of returns object.

→ value can be added or removed to an object.

→ Delete values using delete.

Ex-: > Product.company
< mango.

→ console.log (product);

→ console.log (Product.company);

> delete product.comany;
> console.log (product);

(39) **Inside object :→**

```
ex-: let product = {
        company : 'mango',
        itemName : 'cotton striped t-shirt',
        price : 861,
        rating : {
            stars : 4.5,
            no.ofReviews : 87
            },
        displayPrice : function () {
            return `$${this.price.toFixed(2)}`;
            }
        };
```


product.price.toFixed(2)}`;

→ objects can contain primitives like numbers & strings.
→ objects can contain other objects and are called nested object.
→ functions can be object projects.
→ functs inside an object are called methods.
→ Null values : Intentionally leaving a propery empty.

40 Autoboxing :→

1. Automatic Conversion of primitives to objects.
2. Allows properties & methods to be used on primitives.
3. Example: Strings howe properties and methods like length, touppercase, etc.

Ex-: > console.log ('I am priyanshu'.length);
<- 14
] + count space as well.

> console.log ('i am priyanshu'.touppercase);
> " " (" " " tolowercase);
<- I AM PRIYANSHU tostring);

Ex-: Console.log ('This is kG coding'.replace ('kG', 'knowledge gate'));

41. Object References

→ objects work based on references, not actual data.
→ copying an object copies the reference, not the actual object.
→ when comparing with ==, you are comparing references, not content.
→ Changes to one reference affects all copies.

Ex-:
```
let a = 5;
let b = a;
Console.log (`a=${a}, b=${b}`);
a = 8;
Console.log (`a=${a}, b=${b}`);
```

<- a=5, b=5
a=8, b=5

Ex:- let p = { pop: 'Rajesh' };
let q = { pop: 'Rajesh' };
console.log(p === q);
> false

→ To compare the two objects whether same or not.

42. Object shortcuts

→ De-structuring : extract properties from objects easily.
→ we can extract more than one property at once.
→ shorthand property : { message : message }
Simplifies to just message.
→ shorthand method : Define methods directly inside the object without the function keyword.

// Destructuring :→

```
let product = {
company : 'mango',
itemName : 'cotton striped t-shirt',
Price : 861
  };
let company = product.company
  let { company } = product;
```

// Property shorthand :-

```
let price = 861;
 let product = {
  Company : 'mango'
itemName : 'cotton striped t-shirt',
  price : price
 };
let product1 = {
   Company : 'mango',
  itemName : 'cotton striped t-shirt'
   Price
    };
```

// method shorthand :

```
let product = {
 Company : 'mango',
 itemName : 'cotton striped t-shirt',
 displayprice : function ( ) {
      return `$${this.price.toFixed(2)}`;
   } };
 let product1 = {
      company : 'mango'
      item-name : 'cotton-tee'
```

displayprice( ) {
return `$${this.price.toFixed(2)}`;

5

```
let sum2 = (num1, num2) => num1 + num2;
let square = num => num * num; (9)
```

* practice exercise (objects) :→

1. create object to represent a product from myntra.
2. create an object with two references and log changes to one object by changing the other one.
3. use bracket notation to display delivery-time.
4. Given an object {message: 'good job', status: 'Complete'}, use de-structuring to create two variables message and status.
5. Add function isIdenticalProduct to compare two product objects.

41. object references

1. ~~Objects~~ work based on references, not actual data.
2. Copying an object copies the references, not the actual object.
3. when comparing with ==, you are comparing references, not content.
4. Changes to one reference affects all copies.

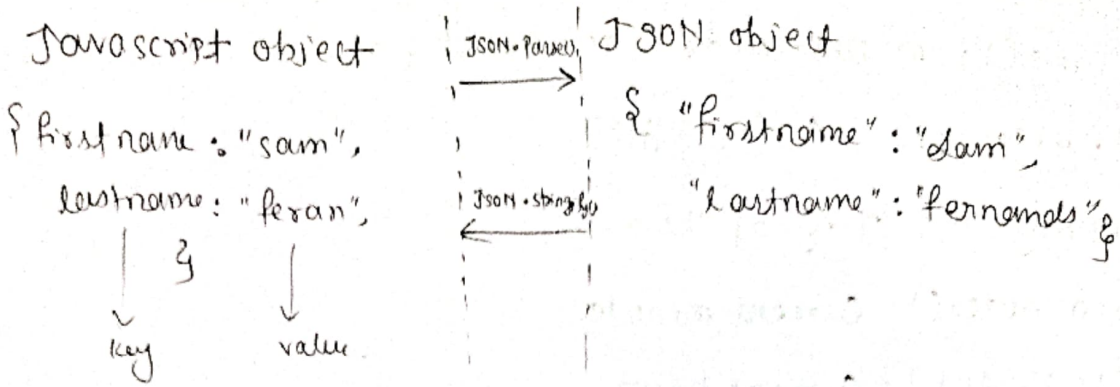
43. JSON, Local storage, Date & DOM

43. What is JSON?
44. Local storage
45. Date
46. Dom properties & methods.

→ Javascript object Notation : Not the same as JS object, but similar.
→ Common in network calls & data storage.
→ JSON.stringify() and JSON.parse()
→ strings are easy to transport over network.
→ JSON requires double quotes. escaped as \".
→ JSON is data format, JS object is a data structure.

let square = num => num * num; (9)

Ex-: Javascript object



44. Local Storage

- Persistent data storage in the browser.
- SetItem : Stores data as key-value pairs.
- only strings can be stored
- getItem : Retrieves data based on key.
- Other methods : localstorage.clear(), removeItem().
- Do not store sensitive information viewable in Storage Console.

Ex-:
```
localStorage.~~setting~~ setItem
    "user",
    JSON.stringify({
        name: "Gopi Gorantala"
        age : 32});
        );
Const user = JSON.parse(
        localStorage.getItem("user")
        );
```

Ex-: localStorage.setItem('name', 'kaCoding');
localStorage.setItem('price', '3456');

45. Date

1. new data() creates a new date object with the current date & time.
2. Key methods : →

let square = num => num * num; (6) (9)

- getTime(): milliseconds since Epoch.
- getFullYear(): 4-digit year
- getDay(): Day of the week
- getMinutes(): Current minute
- getHourse(): Current hour.
- Crucial for timestamps, scheduling, etc.

46. DOM properties & methods

DOM and Elemet properties :→

1. location
2. title
3. href
4. domain
5. innerHTML
6. innerText
7. classList

DOM & Elemet methods :→

1. getElemetById()
2. querySelector()
3. classList : add(), remove()
4. createElemet()
5. appendChild()
6. removeChild()
7. replaceChild()

Practice exercise

JSON, Local strage, Date & DOM

① ; Display good morning, afternoon & night based on current hour.

② Add the name to the output too.

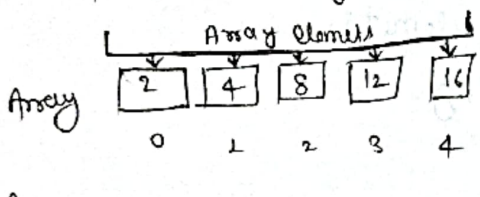
③ create a button which shows the no. how many times it has been pressed.

- Also, it has different colors for when it has been pressed odd or even times.
- The click count should also survive browser refresh.

47. Array & loops

47. What is an Array?
48. Array syntax & values
49. Array properties and methods
50. what is a loop?
51. while loop
52. Do while loop
53. for loop
54. Accumulator pattern
55. Break & Continue

* ① What is an Array?



- An array is just a list of values.
- Index : starts with 0.
- Arrays are used for storing multiple values in a single variable.

Syntax :—

```
let myArray = [1, 'ka coding', null, true, {likes: '1 million'}];
```

- use [ ] to create a new array. [] brackets enclose list of values.
- Array can be saved to a variable.
- Accessing values : use [] with index.
- Syntax rules:-
  - → Brackets start and end the array.
  - → values separated by commas.
  - → Can span multiple lines.
- Arrays can hold any value, including arrays.
- typeof operator on Array returns object.

Ex: let numbers = [1,2,3,4,5]

→ console.log(numbers); ⇒ outputs [1,2,3,4,5]

→ console.log(numbers[4]) ⇒ " 5

49. Array properties & methods

1. Array.isArray() checks if a variable is an array.
2. length property holds the size of the array.
3. Common methods:→
   - push/pop : Add or remove to end
   - shift/unshift : Add or remove from front.
   - splice : Add or remove elemets.
   - tostring : convert to string.
   - sort : sort elmets.
   - valueOf : Get array itself
4. Arrays also use reference like objects.
5. De-structuring also works for Arrays.

Ex: console.log(Array.isArray(numbers)); → true

ex: console.log(numbers.length);

Ex: arr.push(5);

Ex: arr.pop(3);

Ex: let arr = [3,5,7,6]

console.splice(1,2);

→ output [3,6]

Ex: push/unshift

↓ add at the end

↓ add at the beginning

Ex: arr.log(arr.tostring());

~~Ex:~~

ex: arr.sord();

ex: arr.valueof(); } output :- [3,5,6,7].

let square = num => num * num; (9)

→ References: —

Ex: let arr2 = arr;

```
arr2[0] = 22;
console.log(arr);
```

50. What is loop?

1- Code that runs multiple times based on a condition.

2. Loops also alter the flow of execution, similar to functions.

- functions : Reusable blocks of code.
- Loops : Repeated execution of code.

3- Loops automate repetive.

for, while, do while, - — etc.

51. while loop

```
while (condition) {
}
```

Ex:

```
let num = 1;
while (n <= 10) {
    console.log (num);
    num++;
}
```

52. Do while loop

```
do {
}
while (condition);
```

Ex:-

```
let n=0
do {
    console.log(n);
    n++;
}
while (n<5);
```



```
let square = num => num * num; (9)
```

53. For loop

```
for (initialize; condition; update) {
}
```

Ex-:
```
for(let i=0 ; i<10; i++) {
  console.log(i);
}
```

54. Accumulator pattern

- A pattern to accumulate values through looping.
- Common scenarios:
  - Sum all the numbers in an array.
  - Create a modified copy of an array.

55. Break & continue

1. Break lets you stop a loop early, or break out of a loop.
2. Continue is used to skip one iteration or the current iteration.
3. In while loop remember to do the increment manually before using continue.

Ex-:
```
let arr = [1,2,3,4,5,6,7,8];
arr1 = arr.slice(1,4);
Console.log(arr1);
```
> [2,3,4]

Q A no. is prime or not?

→ <script>
```
function isprime(num){
    for(let i=2; i<num ; i++){
      if(num%i===0){
          return false;
      }
    }
console.log(isprime(5));
```
> true

Q In array [1,-6,5,7] print positive no.

→
```
let arr[1,-6,5,7,-98]
for(let index=0; index<=4
                  index++){
if(arr[index]<0){
  continue;
}
else{
 Console.log(arr[index]);
}
}
```

Project → To do list and Advance function

56 Anonymous functions As values

57 Arrow functions

58 setTimeout & setItems

59 Event listener

60 for each loop

61 Array methods.

56 Anonymous function As values :→

```
let sum = function (num1, num2) {
        return num1 + num2;
};
```

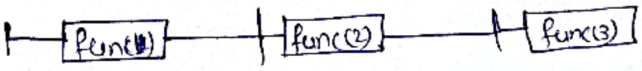
1. functions in javascript are first-class citizens ; they can be assigned to variables.
2. functions defined without a name, often assigned to a variable.
3. Anonymous functions can be properties in objects.
4. can be passed as arguments to other functions.
5. Invoked using () after the function name or variable.
6. console.log(myfunction); and typeof myfunction will both include it's a function.

57 Arrow functions :→

```
let sum = function (num1, num2) {
        return num1 + num2;
};
let sum1 = (num1, num2) => {
        return num1 + num2;
};
let sum2 = (num1, num2) => num1 + num2;
let square = num => num * num; (9)
```

1. A concise way to write anonymous functions.
2. for single Argument: Round brackets optional.
3. for single line: curly brackets & return optional.
4. often used when passing function as Arguments.

58. setTimeout & setInterval



1. functions for executing code asynchronously after a delay.
2. setTimeout runs once; setInterval runs repeatedly setTimeout
3. setTimeout:
   - syntax: setTimeout(function, time)
   - cancel: clearTimeout(timerID)
4. setInterval:
   - syntax: setInterval(function, time)
   - cancel: clearInterval(intervalID)

59. Event Listener

```
const button = document.querySelector(".btn")
button.addEventListner("click", event => {
        console.log("Hello!");
})
```

1. what is an event: occurrence like clicks, mouse movement, keyboard input (e.g., birthday, functions).
2. using querySelector to attach listeners.
3. multiple listeners: you can add more than one.
4. removeEventListener('event', functionVariable);

60. for each loop

```
let foods = ['bread', 'rice', 'meat', 'pizza'];
foods.forEach(function(food){
    console.log(food);
})
```

1. A method for array iteration, often preferred for readability.
2. parameters: one for item, optional second for index.
3. using return is similar to continue in traditional loops.
4. Not straightforward to break out of a forEach loop.
5. when you need to perform an action on each array element and don't need to break early.

61. Array methods

[- , — , --] . map(cook) ⇒ [- - , - - , - -]
[- , — , --] . filter(isVegetarian) ⇒ [— , —]

1. filter method :→
   - syntax : arr.filter((value, index) => return true/false)
   - use : filters elements based on condition.
2. map method :→
   - syntax : array.map((value) => return newValue)
   - use : Transforms each element.

Ex:- let arr = [1,2,3,4,5,6,7,8,9];

```
let odds = arr.filter((num, index) => {
    return true;
} else {
    return false;
});
```

```
let newodds = arr.filter (
    (num, indx) => num % 2 === 1
);
console.log (newodds);
```

practice exercise

Advance functions

1. create a variable multiply and assign a function to this variable that multiplies two numbers. call this method from the variable.
2. Create a function runTwice that takes a function as a parameter and then runs that method twice.
3. create a button which should grow double in size on clicking after 2 seconds.
4. In the above exercise add the behavior using an event listener instead of onclick.
5. create a function that sums an array of numbers. Do this using a for-each loop.
6. create a function that takes an array of numbers and return their squares map function.

=

Project :→ Myntra Web.

Complete Javascript