// <<< 📘 Javascript Books Chapters >>>

// ---

// ========= <<< 🔷Chapters Summary >>> =========

// 1. Alerts

// Definition: user ko message show karne ke liye ek simple popup box.

// Syntax:

// alert("Your message here");

// Notes:

// execution ko temporarily stop kar deta hai jab tak user "OK" press na kare.

// sirf informational messages ke liye use hota hai.

// 2. Variables for Strings

// Definition: string matlab text ko store karna variable me.

// Syntax:

// let name = "Hasnain";

// Important:

// String always quotes me hoti hai " " ya ' '.

// Template literals: backticks (`Hello ${name}`).

// 3. Variables for Numbers

// Definition: numbers ko store karna variable me.

// Syntax:

// let age = 22;

// let price = 99.5;

// Important:

// Integers aur decimals dono store karte hain.

// Numbers ke sath math operators use hote hain.

// 4. Variable Names: Legal and Illegal

// Legal:

// letters, numbers, _, $

// cannot start with number

// case-sensitive (name ≠ Name)

// Illegal:

// space in name (user name ❌)

// reserved keywords (var, let, if ❌)

// special characters (@, %, - ❌)

// 5. Math Expressions: Familiar Operators

// Definition: normal math operators

// Operators:

// + (addition)

```
// - (subtraction)

// * (multiplication)

// / (division)


// 6. Math Expressions: Unfamiliar Operators

// Operators:

// % → modulus (remainder)

// ++ → increment (by 1)

// -- → decrement (by 1)

// ** → exponentiation (power)


// 7. Math Expressions: Eliminating Ambiguity

// Definition: order of operations clear karna with parentheses.


// Notes:

// JS follows BODMAS/PEMDAS rules.

// Example order: () → ** → * / % → + -


// 8. Concatenating Text Strings

// Definition: 2 ya zyada strings ko jodna.


// Operators/Methods:

// + → "Hello " + "World"

// Template literal → `Hello ${name}`

// .concat() method
// 9. Prompts
// Definition: user se input lene ke liye popup box.

// Syntax:
// let user = prompt("Enter your name:");

// Notes:
// Returns string hamesha.
// Agar cancel ho to null return karta hai.

// 10. if Statements
// Definition: condition check karne ke liye use hota hai.

// Syntax:
// if (condition) {
//    // run code
// } else if (anotherCondition) {
//    // run code
// } else {
//    // run code
// }

// Important:
// Condition hamesha boolean (true/false) return kare.
// Comparison operators: ==, ===, !=, <, >, <=, >=
// Logical operators: && (AND), || (OR), ! (NOT)
// 11. Comparison Operators
// Definition: values ko compare karne ke liye.

// Operators:
// == → equal (type convert karta hai)
// === → strict equal (type + value check)
// != → not equal
// !== → strict not equal
// <, >, <=, >= → less/greater comparisons

// 12. if...else and else if Statements
// Definition: conditional branching.

// Syntax:
// if (condition) { }
// else if (another) { }
// else { }

// Notes:
// multiple conditions check karne ke liye useful.
// else block optional hota hai.

// 13. Testing Sets of Conditions
// Definition: ek se zyada conditions ko test karna ek hi statement me.
```

// Operators:

// && → AND (all conditions true)

// || → OR (at least one true)

// ! → NOT (reverse condition)

// 14. if Statements Nested

// Definition: ek if ke andar dusra if.

// Use: jab complex conditions ho jo step-by-step test karni ho.

// Note: zyada nested if se code complex ho jata hai → try to simplify with logical operators.

// 15. Arrays

// Definition: ek hi variable me multiple values store karne ka tarika.

// Syntax:

// let arr = [10, 20, 30];

// Important Methods:

// .length → number of elements

// index access (arr[0])

// 16. Arrays: Adding and Removing Elements

// Adding:

// .push() → end me add

// .unshift() → start me add

// Removing:

// .pop() → end se remove

// .shift() → start se remove

// 17. Arrays: Removing, Inserting, and Extracting Elements

// Important Methods:

// .splice(start, deleteCount, items...) → remove/insert anywhere

// .slice(start, end) → extract (new array banata hai)

// 18. for Loops

// Definition: repeat karne ka control structure.

// Syntax:

// for (let i = 0; i < 5; i++) { }

// Parts:

// init → let i = 0

// condition → i < 5

// update → i++

// 19. for Loops: Flags, Booleans, Array length, and Breaks

// Flags: ek variable jo loop ke andar condition track kare.

// Booleans: true/false values check karne ke liye.

// Array length: loop ka end decide karne ke liye .length.

// Break: loop ko forcefully stop karna.

// 20. for Loops Nested

// Definition: ek loop ke andar doosra loop.

// Use: jab 2D arrays, tables, ya combinations banane ho.

// Note: bahut heavy processing kar sakte hain → optimize zaroor karo.

// 21. Changing Case

// Definition: string ko uppercase/lowercase me convert karna.

// Methods:

// .toUpperCase() → sab capital letters.

// .toLowerCase() → sab small letters.

// 22. Strings: Measuring Length and Extracting Parts

// Measuring Length:

// .length → total characters count.

// Extracting Parts:

// .slice(start, end) → substring extract karta hai.

// .substring(start, end) → similar to slice.

// .substr(start, length) → start index + kitne chars chahiye.

// 23. Strings: Finding Segments

// Definition: ek segment (word/phrase) ko string me search karna.

// Methods:

// .indexOf("word") → first match index.

// .lastIndexOf("word") → last match index.

// .includes("word") → true/false return.

// .startsWith("abc"), .endsWith("xyz") → check karna beginning/end.

// 24. Strings: Finding a Character at a Location

// Methods:

// .charAt(index) → ek character return.

// str[index] → bracket notation se direct access.

// .charCodeAt(index) → unicode value return.

// 25. Strings: Replacing Characters

// Definition: string ke andar text replace karna.

// Method:

// .replace("old", "new") → first match replace.

// .replaceAll("old", "new") → sab replace.

// regex bhi use hota hai: .replace(/old/g, "new").

// 26. Rounding Numbers

// Methods:

// Math.round(num) → nearest integer.

// Math.floor(num) → always down.

// Math.ceil(num) → always up.

// Math.trunc(num) → remove decimal part.

// 27. Generating Random Numbers

// Method:

// Math.random() → 0 se 1 ke beech decimal number.

// Custom Range:

// Math.floor(Math.random() * 10) // 0–9

// Math.floor(Math.random() * 100) + 1 // 1–100

// 28. Converting Strings to Integers and Decimals

// Methods:

// parseInt("123") → integer.

// parseFloat("12.34") → decimal number.

// Note: agar string number se start hoti hai to parse kar lega.

// 29. Converting Strings to Numbers, Numbers to Strings

// Strings → Numbers:

// Number("123")

// + "123" (unary plus operator)

// Numbers → Strings:

// .toString()

// String(123)

// 30. Controlling the Length of Decimals

// Methods:

// .toFixed(n) → fixed decimal places (string return karta hai).

// .toPrecision(n) → total digits (string return karta hai).

```
// Example:
// let num = 3.14159;
// num.toFixed(2);     // "3.14"
// num.toPrecision(3); // "3.14"

// 31. Getting the Current Date and Time
// Definition: JavaScript Date object se current date/time milta hai.

// Syntax:
// let now = new Date();

// Note: returns full date + time of user’s system.

// 32. Extracting Parts of the Date and Time
// Methods:
// .getFullYear() → year (2025)
// .getMonth() → month (0–11)
// .getDate() → day of month (1–31)
// .getDay() → day of week (0–6)
// .getHours(), .getMinutes(), .getSeconds(), .getMilliseconds()

// 33. Specifying a Date and Time
// Definition: custom date/time banana.

// Syntax:
```

```
// let d1 = new Date("2025-09-12");

// let d2 = new Date(2025, 8, 12, 10, 30, 0); // year, month(0–11), day, hr, min, sec


// 34. Changing Elements of a Date and Time

// Methods:

// .setFullYear(year)

// .setMonth(month)

// .setDate(day)

// .setHours(hr), .setMinutes(min), .setSeconds(sec)


// Note: directly modify kar deta hai date object ko.


// 35. Functions

// Definition: reusable code block.


// Syntax:

// function greet() {

//   console.log("Hello");

// }


// Notes: DRY principle (Don't Repeat Yourself).


// 36. Functions: Passing Them Data

// Definition: parameters ke zariye data bhejna.


// Syntax:
```

```
// function greet(name) {
//   console.log("Hello " + name);
// }

// Note: arguments function call ke waqt diye jate hain.

// 37. Functions: Passing Data Back from Them
// Definition: return statement se function output deta hai.

// Syntax:
// function add(a, b) {
//   return a + b;
// }

// Note: return ke baad function execution ruk jata hai.

// 38. Functions: Local vs. Global Variables
// Global: function ke bahar declared → sab jagah accessible.
// Local: function ke andar declared → sirf usi function ke andar accessible.
// Note: same name ke variables local ko preference dete hain.

// 39. switch Statements: How to Start Them
// Definition: multiple conditions handle karne ka alternate to if...else.

// Syntax Start:
// switch (expression) {
```

```
//   case value1:
//     // code
//     break;
// }

// 40. switch Statements: How to Complete Them
// Important Parts:
// case → match value
// break → exit case
// default → fallback if no match

// Note: bina break ke cases fall-through karte hain.

// 41. while Loops
// Definition: condition true rahe to code repeat hota hai.

// Syntax:
// while (condition) {
//   // code
// }

// Note: agar condition kabhi false na ho to infinite loop ban sakta hai.

// 42. do...while Loops
// Definition: pehle code run hoga, phir condition check hogi.
```

// Syntax:

// do {

//   // code

// } while (condition);

// Note: at least 1 bar run hamesha hoga.

// 43. Placing Scripts

// Definition: HTML me JavaScript ko place karna.

// Ways:

// <head> → page load hone se pehle (blocking).

// <body> end → recommended (page load fast).

// external .js file via <script src="app.js"></script>.

// 44. Commenting

// Definition: code explain karne ya disable karne ke liye.

// Types:

// Single-line: // this is comment

// Multi-line:

// /* this is

//   multi-line comment */

// 45. Events: Link

// Definition: links (anchor tags) ke sath events.

// Examples:

// onclick → jab link click ho.

// onmouseover, onmouseout → hover effects.

// 46. Events: Button

// Definition: button click pe event trigger.

// Examples:

// onclick → form submit ya custom action.

// ondblclick → double-click action.

// 47. Events: Mouse

// Definition: mouse related actions capture karna.

// Important Events:

// onclick, ondblclick

// onmouseover, onmouseout

// onmousemove, onmousedown, onmouseup

// 48. Events: Fields

// Definition: form fields pe events.

// Important Events:

// onfocus → jab input focus ho.

// onblur → jab input lose focus kare.

// onchange → jab value change ho aur focus leave ho.

// oninput → jab user type kare.

// 49. Reading Field Values

// Definition: input ya form field ki value read karna.

// Syntax:

// let val = document.getElementById("username").value;

// 50. Setting Field Values

// Definition: input field ki value set/update karna.

// Syntax:

// document.getElementById("username").value = "Hasnain";

// 51. Reading and Setting Paragraph Text

// Definition: paragraph (<p>) ka text lena ya update karna.

// Syntax:

// let text = document.getElementById("para1").innerText;   // read

// document.getElementById("para1").innerText = "New text"; // set

// Note: innerText ya textContent use hota hai.

// 52. Manipulating Images and Text

// Definition: images aur text dynamically change karna.

// Methods:

// .src → image ka source badalna.

// .alt → alternative text.

// .innerHTML / .innerText → text change karna.

// 53. Swapping Images

// Definition: ek image ke jagah dusri image show karna.

// Syntax:

// document.getElementById("pic").src = "image2.jpg";

// Use: hover effects, gallery, sliders.

// 54. Swapping Images and Setting Classes

// Definition: image swap ke sath CSS class bhi change karna.

// Syntax:

// let img = document.getElementById("pic");

// img.src = "image2.jpg";

// img.className = "highlight";

// Note: CSS styling bhi apply kar sakte ho.

// 55. Setting Styles

// Definition: inline CSS ko JS ke zariye set karna.

// Syntax:

// document.getElementById("box").style.color = "red";

// document.getElementById("box").style.backgroundColor = "yellow";

// Note: style.propertyName use hota hai camelCase me.

// 56. Target All Elements by Tag Name

// Definition: ek hi tag ke multiple elements select karna.

// Syntax:

// let paras = document.getElementsByTagName("p");

// Note: returns an HTMLCollection (array-like).

// 57. Target Some Elements by Tag Name

// Definition: collection me se kuch elements target karna.

// Syntax:

// let paras = document.getElementsByTagName("p");

// paras[0].innerText = "Changed first paragraph";

// Note: index se specific element access hota hai.

// 58. The DOM

// Definition: Document Object Model = webpage ka structured tree.

// Root: document object.

// Access Methods:

// getElementById()

// getElementsByTagName()

// querySelector() / querySelectorAll()

// 59. The DOM: Parents and Children

// Definition: DOM me har node ka ek parent aur children hote hain.

// Properties:

// .parentNode → parent element.

// .children → child elements (HTMLCollection).

// .firstChild / .lastChild.

// 60. The DOM: Finding Children

// Definition: specific element ke child nodes find karna.

// Methods:

// .children → only element nodes.

// .childNodes → all nodes (including text, comments).

// .firstElementChild, .lastElementChild.

// 61. The DOM: Junk Artifacts and nodeType

// Definition: DOM me non-element nodes (whitespace, comments) → junk artifacts.

// Property:

// .nodeType → node ka type return karta hai

// 1 → element node

// 3 → text node

// 8 → comment node

// Note: loops me filter karne ke liye useful.

// 62. The DOM: More Ways to Target Elements

// Methods:

// getElementsByClassName("class")

// querySelector("selector") → first match

// querySelectorAll("selector") → all matches

// Note: modern JS me querySelector zyada use hota hai.

// 63. The DOM: Getting a Target's Name

// Definition: element ke tag ya name ko read karna.

// Properties:

// .tagName → element ka HTML tag (DIV, P)

// .name → form element ka name attribute

// .id → element ka id

// 64. The DOM: Counting Elements

// Definition: parent ya page ke andar total elements count karna.

// Methods:

// .length → HTMLCollection ya NodeList ke liye

// .childElementCount → parent ke child elements count

// .children.length → same as above

// .childNodes.length → saare nodes count (junk included)

// 65. The DOM: Attributes

// Definition: element ke attributes ko read, set, remove karna.

// Methods:

// getAttribute("attr")

// setAttribute("attr","value")

// removeAttribute("attr")

// hasAttribute("attr")

// Properties: direct access via .id, .className, .src, etc.

// 66. The DOM: Attribute Names and Values

// Definition: element ke saare attribute names aur unki values access karna.

// Methods:

// .attributes → NamedNodeMap

// .getAttribute("attr")

// .setAttribute("attr", value)

// .removeAttribute("attr")

// .hasAttribute("attr")

// 67. The DOM: Adding Nodes

// Definition: naya element ya text node create kar ke DOM me insert karna.

// Methods:

// document.createElement("tag")

// document.createTextNode("text")

// parent.appendChild(node)

// parent.insertBefore(newNode, existingNode)

// .append() / .prepend()

// 68. The DOM: Inserting Nodes

// Definition: element ko specific jagah insert karna.

// Methods:

// appendChild() → end me

// insertBefore(newNode, referenceNode) → beech me

// .prepend() → start me

// .append(node1, node2, …) → multiple nodes

// .insertAdjacentElement(position, element) → precise location

// 69. Objects

// Definition: key-value pairs ka collection.

```
// Syntax:

// let person = {name: "Hasnain", age: 22};


// Note: properties aur methods store kar sakte ho.


// 70. Objects: Properties

// Definition: objects ke andar values stored as properties.


// Access:

// Dot notation → person.name

// Bracket notation → person["age"]

// Important: properties can be updated, added, or deleted.


// 71. Objects: Methods

// Definition: objects ke andar functions store karna aur call karna.


// Syntax:

// let person = {

//   name: "Hasnain",

//   greet: function() { console.log("Hello " + this.name); }

// };

// person.greet(); // call method


// Note: this keyword se current object reference hota hai.
// 72. Objects: Constructors
// Definition: template ke tarah function jo objects banata hai.

// Syntax:
// function Person(name, age) {
//   this.name = name;
//   this.age = age;
// }
// let p1 = new Person("Ali", 25);

// Note: new keyword use hota hai.

// 73. Objects: Constructors for Methods
// Definition: constructor ke andar methods define karna.

// Syntax:
// function Person(name) {
//   this.name = name;
//   this.greet = function() { console.log("Hello " + this.name); };
// }
// let p1 = new Person("Ali");
// p1.greet();

// Note: each object apna method copy karega.

// 74. Objects: Prototypes
```

// Definition: methods aur properties share karne ka tarika.

// Syntax:

// function Person(name) { this.name = name; }

// Person.prototype.greet = function() { console.log("Hello " + this.name); };

// Note: sab objects same prototype method share karte hain.

// 75. Objects: Checking for Properties and Methods

// Methods:

// obj.hasOwnProperty("prop") → check if property exists

// "prop" in obj → check if property exists (own + prototype)

// typeof obj.method === "function" → check if method exists

// 76. Browser Control: Getting and Setting the URL

// Definition: current page URL ko read ya change karna.

// Properties:

// window.location.href → get or set complete URL

// window.location.protocol → http/https

// window.location.host → domain

// 77. Browser Control: Getting and Setting the URL Another Way

// Methods:

// window.location.assign(url) → navigate to new URL

// window.location.replace(url) → replace current page

// window.location.reload() → reload page

// 78. Browser Control: Forward and Reverse

// Definition: browser history navigate karna.

// Methods:

// window.history.back() → go back

// window.history.forward() → go forward

// window.history.go(n) → n steps in history

// 79. Browser Control: Filling the Window with Content

// Definition: browser window me naya content write karna.

// Methods:

// document.write("text") → page me direct content add

// Note: mostly page load ke time use hota hai, dynamic use risky

// 80. Browser Control: Controlling the Window's Size and Location

// Definition: window size aur position control karna.

// Methods/Properties:

// window.resizeTo(width, height) → set size

// window.resizeBy(dw, dh) → increase/decrease size

// window.moveTo(x, y) → move window

```
// window.moveBy(dx, dy) → relative move


// 81. Browser Control: Testing for Popup Blockers

// Definition: browser me popups allowed hain ya blocked check karna.


// Syntax/Method:

// let popup = window.open("about:blank");

// if (!popup || popup.closed || typeof popup.closed == 'undefined') {

//   alert("Popup blocked");

// }


// Note: modern browsers me auto popups mostly block hote hain.


// 82. Form Validation: Text Fields

// Definition: input text field me valid data check karna.


// Methods/Properties:

// .value → input value

// required attribute → empty na ho

// pattern attribute → regex validation


// 83. Form Validation: Drop-downs

// Definition: select element me valid option select karna.


// Methods/Properties:

// .value → selected option
```

// .selectedIndex → selected position

// .options → all options

// 84. Form Validation: Radio Buttons

// Definition: radio group me ek option select hai ya nahi check karna.

// Methods/Properties:

// radio.checked → true/false

// loop over group to find selected

// 85. Form Validation: ZIP Codes

// Definition: postal code validation.

// Methods:

// regex → /^\d{5}(-\d{4})?$/

// .test() → match check

// 86. Form Validation: Email

// Definition: email format validate karna.

// Methods:

// regex → /^\S+@\S+\.\S+$/

// .test() method

// type="email" in HTML input also helps

// 87. Exceptions: try and catch

```javascript
// Definition: runtime errors handle karna.

// Syntax:
// try {
//   // code
// } catch (error) {
//   // handle error
// }

// Note: program crash se bachata hai.

// 88. Exceptions: throw
// Definition: custom error create karna.

// Syntax:
// if(!age) throw "Age required";

// Note: try...catch ke sath handle karna recommended.

// 89. Handling Events within JavaScript
// Definition: HTML elements ke actions handle karna.

// Common Events:
// Mouse → onclick, ondblclick, onmouseover, onmouseout
// Keyboard → onkeydown, onkeyup, onkeypress
// Forms → onsubmit, onchange, onfocus, onblur
// Window → onload, onresize, onscroll

// Methods:
// Inline → <button onclick="myFunc()">
// JS → element.addEventListener("click", func)

// ------------------------------------------------------------------

// ========= <<< 🔷Examples and Methods >>> =========
// --------------------------------------------------------------
// basic data types - (string, number, boolean, undefined, null)
// --------------------------------------------------------------

// 1
// var name = "ahmed";
// console.log(name)
// console.log(typeof name)
// console.log("==============");

// 2
// var age = 20;
// console.log(age)
// console.log(typeof age)
// console.log("==============");

// 3
// var isAdult = true;

// console.log(isAdult)

// console.log(typeof isAdult)

// console.log("==============");


// 4

// var x;

// console.log(x)

// console.log(typeof x)

// console.log("==============");


// 5

// var y = null;

// console.log(y)

// console.log(typeof y)

// console.log("==============");


// ------------------

// Chapter # 1 Alerts

// ------------------


// alert("Error! Please enter a valid password")

// alert("Welcome to JS Land... \nHappy Coding!")

// alert("Welcome to JS Land...")

// alert("Happy Coding!")

// alert("Hello... I can run JS through my web browser's console")
// --------------------------------
// Chapter # 2 Variables for Strings
// --------------------------------

// var myName = "Jhone Doe"
// var myAge  = "15 years old"
// var myData = "Certified Mobile Application Development"
// var email  = "example@example.com"
// var book   = "'A smarter way to learn JavaScript'"

// console.log(
//     typeof myName ,
//     typeof myAge  ,
//     typeof myData ,
//     typeof email  ,
//     typeof book
// );

// --------------------------------
// Chapter # 3 Variables for Numbers
// --------------------------------

// var age        = 15
// var visitTimes = 14
// var birthYear  = 2000
// console.log(
//     typeof age ,
//     typeof visitTimes  ,
//     typeof birthYear
// );

// --------------------------------------------
// Chapter # 4 Variable Names Legal and Illegal
// --------------------------------------------

// Legal Variables
// var firstVariable = ""
// var first_variable= ""
// var $variable     = ""
// var variable12    = ""
// var $variable_1   = ""

// illegal Variables
// var 12variable = "" (cannot start with a number)
// var 123        = "" (only number is not allowed)
// var (variable) = "" ( [()] bracket is not allowed)
// var @variable  = "" (special characters [@,!,&,-] are not allowed)
// var if         = "" (reserved keyword)

// ------------------------------------------------
// Chapter # 5 Math Expressions: familiar operators

// -----------------------------------------------


// 1

// var a = 5;

// var b = 5;

// var sum = a + b;

// console.log(sum);


// 2

// var a = 5;

// var b = a;

// var sum = a + b;

// console.log(sum);


// 3

// var a = 4;

// var b = a + 2;

// var sum = a + b;

// console.log(sum);


// 4

// var a = 5;

// var b = 5;


// console.log("Add : " , a + b);
// console.log("Sub : " , a - b);

// console.log("Mul : " , a * b);

// console.log("Div : " , a / b);

// console.log("Mod : " , a % b);


// 5

// Operators Problems [BDMAS]

// console.log(2 * (4 + 10));

// console.log(2**4)

// console.log((2 + 10) * 2 ** 2 ** 2 + 4 / 2);

// console.log(12 * 2 ** 4 + 4 / 2);

// console.log(12 * 16 + 4 / 2);


// 6

// var a = 10;

// var b = "20";

// var c = "ahmed";


// console.log(a + b) // 1020

// console.log(a + Number(b)) // 30

// // Number() method : covert string to number ["1" => 1]

// console.log(a - b) // -10

// console.log(a * b) // 200

// console.log(a / b) // 0.5

// console.log(a * c) // NaN
// Task 1 (Basic Calculator)
// var num1 = Number(prompt("Enter any number used for all"))
// var num2 = Number(prompt("Enter any number used for all"))

// var add = num1 + num2
// var sub = num1 - num2
// var mul = num1 * num2
// var div = num1 / num2
// var mod = num1 % num2

// console.log(`1. Addition of ${num1} and ${num2} is ${add}`)
// console.log(`2. subtraction of ${num1} and ${num2} is ${sub}`)
// console.log(`3. Multiplication of ${num1} and ${num2} is ${mul}`)
// console.log(`4. Division of ${num1} and ${num2} is ${div}`)
// console.log(`5. Modulus of ${num1} and ${num2} is ${mod}`)

// Task 2 (Perc-Grade-check)
// var Studentname = +prompt ('Enter Studentname')
// var a = +prompt ('Enter English Mark')
// var b = +prompt ('Enter Urdu Mark')
// var c = +prompt ('Enter Islamiat Mark')
// var d = +prompt ('Enter Math Mark')
// var e = +prompt ('Enter PST Mark')
// var Total = (a+b+c+d+e)

// document.writeln('|Total = 600/ ')

// document.writeln(Total)


// var perc = (Total/500)*100

// document.writeln('|Percentage = ')

// document.writeln(perc)


// Task 3 (Calculator with Border)

// var num1 = +prompt("Enter any value 1")

// var num2 = +prompt("Enter any value 2")

// var add = num1 + num2

// var sub = num1 - num2

// var mul = num1 * num2

// var div = num1 / num2

// var mod = num1 % num2


// document.writeln(

//     "<table width='152px' border='1px'>"+

//     "<tr>"+

//     "<th>"+"ADDITION"+"</th>"

//     +"</tr>"+

//     "<tr>"+

//     "<td>"+add+"</td>"

//     +"</tr>"

//     +"</table>"+
//    "<table width='152px' border='1px'>"+
//    "<tr>"+
//    "<th>"+"SUBSTRACTION"+"</th>"
//    +"</tr>"+
//    "<tr>"+
//    "<td>"+sub+"</td>"
//    +"</tr>"
//    +"</table>"+

//    "<table width='152px' border='1px'>"+
//    "<tr>"+
//    "<th>"+"MULTIPLICATION"+"</th>"
//    +"</tr>"+
//    "<tr>"+
//    "<td>"+mul+"</td>"
//    +"</tr>"
//    +"</table>"+

//    "<table width='152px' border='1px'>"+
//    "<tr>"+
//    "<th>"+"DIVISION"+"</th>"
//    +"</tr>"+
//    "<tr>"+
//    "<td>"+div+"</td>"
//    +"</tr>"
//    +"</table>"+

//    "<table width='152px' border='1px'>"+
//    "<tr>"+
//    "<th>"+"MODULUS"+"</th>"
//    +"</tr>"+
//    "<tr>"+
//    "<td>"+mod+"</td>"
//    +"</tr>"
//    +"</table>"
// )

// ------------------------------------------------
// Chapter # 6 Math Expressions: unfamiliar operators
// ------------------------------------------------

// 1
// let a = 2
// let res1 = a++
// console.log("Post-Increment" , res1);
// let res2 = ++a
// console.log("Pre-Increment"  , res2);

// 2
// let b = 2
// let res3 = b--
// console.log("Pre-Decrement" , res3);

// let res4 = --b

// console.log("Post-Decrement"  , res4);


// 3

// var num1 = 6

// var num2 = 3

// var res = num1++ + num1++ + ++num2 + num2++ + num1++

//  //       6    +   7    +    5   +   3    +   7 + 1

// //              29

// document.write(res)


// 4

// var a = 4

// var b = 2

// var c = 10

// var res = a++ + ++a - --b + c++ + c++ + ++a + a

//        4  +  6  -  1  +  10 +  11 +  7  + 7

//                 9    +   35

//                      44

// document.write(res)


// 5

// var y = 10

// var u = 5

// var w = 2
// var res = y - w * (y + w) + y
// //          -4
// document.write(res)


// ---------------------------------------------------
// Chapter # 7 Math Expressions: eliminating ambiguity
// ---------------------------------------------------


// console.log(2 * (4 + 10));
// console.log(2**4)
// console.log((2 + 10) * 2 ** 2 ** 2 + 4 / 2);
// console.log(12 * 2 ** 4 + 4 / 2);
// console.log(12 * 16 + 4 / 2);


// -------------------------------------
// Chapter # 8 Concatenating text strings
// -------------------------------------


// let firstName = "John"
// let lastName  = "Doe"


// console.log(firstName + " " + lastName);


// -------------------
// Chapter # 9 Prompts
// -------------------
// 1
// prompt("Hello...")
// prompt("How Are you ?")

// 2
// var name = prompt("Please enter your name");
// var age = prompt("Please enter your age");

// console.log(typeof name);
// alert("Your name is " + name);

// console.log(typeof age);
// alert("Your age is " + (Number(age) + 10));

// 3
// let userNum1 = +prompt("Enter first value")
// let userNum2 = +prompt("Enter Second value")
// // + : convert string to number ["1" => 1]

// console.log(userNum1 + userNum2);
// // console.log(userNum1 - userNum2);
// // console.log(userNum1 * userNum2);
// // console.log(userNum1 / userNum2);
// // console.log(userNum1 % userNum2);
// --------------------------
// Chapter # 10 if statements
// --------------------------

// 1 (syntax)
// if () {}

// 2
// let a = 12
// let b = 10

// if (a > b) {
//     console.log("A is greater than B");
// }
// else {
//     console.log("A is less than B");
// }

// 3 (Task)
// link : https://github.com/MuhammadHasnain02/MWD-JS-Assign-2

// --------------------------------
// Chapter # 11 Comparison operators
// --------------------------------

// 1 (conditional statement)
// var x = true;
// if (x) {
//   console.log("TRUE");
// }
// else {
//   console.log("FALSE");
// }

// 2 (> = Greater than operator)
// let a = 12
// let b = 10

// if (a > b) {
//     console.log("A is greater than B");
// }
// else {
//     console.log("A is less than B");
// }

// 3 (< = Less than operator)
// let a = 12
// let b = 10

// if (a < b) {
//     console.log("A is greater than B");
// }
// else {
//     console.log("A is less than B");
// }

// 4 (>= = Greater than equal to operator)
// let a = 10
// let b = 10

// if (a >= b) {
//     console.log("A is greater than B");
// }
// else {
//     console.log("A is less than B");
// }

// 5 (<= = Less than equal to operator)
// let a = 10
// let b = 10

// if (a <= b) {
//     console.log("A is greater than B");
// }
// else {
//     console.log("A is less than B");
// }
// 6 (Equal to operator)

// let a = 10

// let b = 10


// if (a == b) {

//     console.log("A is equal to B");

// }

// else {

//     console.log("A is not equal to B");

// }


// 7 (Not Equal to operator)

// let a = 12

// let b = 10


// if (a != b) {

//     console.log("A is Not equal to B");

// }

// else {

//     console.log("A is Equal to B");

// }


// 8 (Not Equal to Equal to operator)

// let a = 10

// let b = "10"

// !== = value and type check
// if (a !== b) {
//     console.log("Yes");
// }
// else {
//     console.log("No");
// }

// 9 (lose equality)
// let a = 10
// let b = "10"
// // [==] = only value check

// if (a == b) {
//     console.log("Yes");
// }
// else {
//     console.log("No");
// }

// 10 (strict equality)
// let a = 10
// let b = "10"
// // [===] = value and type check

// if (a === b) {
```

```
//     console.log("Yes");
// }
// else {
//     console.log("No");
// }

// ---------------------------------------------
// Chapter # 12 if , else and else if statements
// ---------------------------------------------

// All Exercises link : https://github.com/MuhammadHasnain02/MWD-JS-Assign-2

// --------------------------------------
// Chapter # 13 Testing sets of conditions
// --------------------------------------

// 1 (logical operators)
// !   not
// ||  OR
// &&  AND

// let condition = true
// console.log(!condition);

// var name = "ahmed";
// var marks = 60;
// if (name === "ahmed" || marks >= 60) {
//   console.log("TRUE");
// } else {
//   console.log("FALSE");
// }

// if (name === "ahmed" && marks >= 60) {
//   console.log("TRUE");
// } else {
//   console.log("FALSE");
// }

// 2
// let email = "abc@gmail.com"
// let passw = "12345"

// if (email = "abc@gmail.com" && passw == "12345") {
//     console.log("You are successfully login");
// }
// else {
//     console.log("Your Email or password is wrong");
// }

// 3
// All Exercises link : https://github.com/MuhammadHasnain02/MWD-JS-Assign-2
// --------------------------------
// Chapter # 14 if statements nested
// --------------------------------

// 1
// var userValue = prompt("Enter number")

// if(userValue > 0){
//     alert("The numer is positive")
// }
// else if(userValue < 0){
//     alert("The numer is negative")
// }
// else{
//     alert("The numer is zero")
// }

// 2
// if(perc >= 80){
// console.log('|Grade A1|')
// }
// else if(perc >= 70){
// console.log('|Grade A|')
// }
// else if(perc >= 60){
```

```
// console.log('|Grade B|')
// }
// else if(perc >= 50){
// console.log('|Grade C|')
// }
// else if(perc >= 40){
// console.log('|Grade D|')
// }
// else{
// console.log('|Fail|')
// }

// All Exercises link : https://github.com/MuhammadHasnain02/MWD-JS-Assign-2

// -------------------
// Chapter # 15 Arrays
// -------------------

// 1 (syntax)
// let array = [Any Data (string , Number , Boolean etc)]

// 2
// var arr = ['Has','Ali',23,true]
// console.log(arr[0]+' '+arr[1])

// ------------------------------------------------
// Chapter # 16 Arrays: adding and removing elements
// -----------------------------------------------

// 1
// var arr = ['Ali' , 23 , true]
// arr.push('hello')
// console.log(arr)

// 2
// var arr = ['Ali' , 23 , true]
// arr.pop()
// console.log(arr)

// 3
// var arr = ['Ali' , 23 , true]
// arr.unshift('hello')
// console.log(arr)

// 4
// var arr = ['Ali' , 23 , true]
// arr.shift()
// console.log(arr)

// --------------------------------------------------------------
// Chapter # 17 Arrays: removing, inserting, and extracting elements
// --------------------------------------------------------------
// 1 (Adding Index , deleting count and new string)
// var arr = ['Ali' , 23 , true]
// arr.splice(1 , 1) // starting and delete count
// // output ['Ali', true]
// console.log(arr);

// 2
// var arr = ['Ali' , 23 , true]
// arr.splice(1 , 1 , "hello") // starting , delete count and string
// // output ['Ali', 'hh', true]
// console.log(arr);

// 3 (copy array items)
// var arr = ['Ali' , 23 , true]
// let newArr = arr.slice(2 , 3) // starting and ending count
// // output [true]
// console.log(arr);
// console.log(newArr);

// ----------------------
// Chapter # 18 for loops
// ----------------------

// 1 (syntax)
// for ("initialized | condition | increment or decrement") {"body"}
// 2
// for (var i = 1; i <= 10; i++) {
//     console.log(i);
// }

// 3 Infinite Loop
// for (var i = 1; i > 0; i++) {
//     console.log(i);
// }

// 4 (Loop help print Table)
// let userTable = prompt("Enter Table Name")
// let userCount = prompt("Enter Table Count")

// for (let i = 0; i <= userCount; i++) {
//     console.log(`${userTable} * ${i} = ${userTable * i}`);
// }

// 5
// var userinput = prompt('Enter Country Name [First Letter Capital]')
// var arr  = ['Pakistan','India','China','Iran','Iraq']
// var match = false

// for (var i = 0; i < arr.length; i++) {
```

```
//     if(userinput.toLowerCase == arr[i].toLowerCase){
//         match = true
//         alert('Country Found')
//         break
//     }

// }
// if (match == false) {
//     alert('Country Not Found [Put Your Country in Code]')
// }

// ---------------------------------------------------------------
// Chapter # 19 for loops: flags, Booleans, array length, and breaks
// ---------------------------------------------------------------

// 1
// var matchFound = "no";

// for (var i = 0; i <= 4; i++){
//     if (cityToCheck === cleanestCities[i]) {
//         matchFound = "yes";
//         alert("It's one of the cleanest cities");
//     }
// }

// if (matchFound === "no") {
```

```
//     alert("It's not on the list");
// }

// 2
// var numElements = cleanestCities.length;
// var matchFound = false;
// for (var i = 0; i < numElements; i++){
//     if (cityToCheck === cleanestCities[i]) {
//         matchFound = true;
//         alert("It's one of the cleanest cities");
//         break;
//     }
// }
// if (matchFound === false) {
//     alert("It's not on the list");
// }

// -----------------------------
// Chapter # 20 for loops nested
// -----------------------------

// 1
// for (var i = 1; i <= 100; i = i + 10) {

//   for(var j = i; j < i + 10; j++){
```

```
//     document.writeln(j + ' ')

//   }

//   document.writeln('<br>')
// }

// ----------------------------------------
// Chapter # 21 Changing case
// ----------------------------------------

// 1
// var cityToCheck = prompt("Enter your city");
// cityToCheck = cityToCheck.toLowerCase();
// var cleanestCities = ["cheyenne", "santa fe", "tucson", "great falls", "honolulu"];

// for (var i = 0; i <= 4; i++) {
//     if (cityToCheck === cleanestCities[i]) {
//         alert("It's one of the cleanest cities");
//     }
// }

// 2
// let userInput = "AdMiN";
// let correctUsername = "admin";
// if (userInput.toLowerCase() === correctUsername.toLowerCase()) {
//   console.log("Login successful!");
// }
// else {
//   console.log("Username incorrect.");
// }

// --------------------------------------
// Chapter # 22 Strings: Measuring length and extracting parts
// --------------------------------------

// 1 [Measuring length]
// let message = "JavaScript is awesome!";
// console.log(message.length);  // Output: 23

// 2
// let array = [1 , 2 , 3]
// console.log(array.length);

// 3
// let text = "Hello, Hasnain!";
// let part = text.slice(7, 14);
// console.log(part);  // Output: "Hasnain"

// 4
// let str = "JavaScript";
// let part = str.substring(4, 10);
// console.log(part);  // Output: "Script"

// ---------------------------------------
// Chapter # 23 Strings: Finding segments
// ---------------------------------------

// 1 [indexOf()]
// let sentence = "I love JavaScript and JavaScript loves me.";
// let index = sentence.indexOf("JavaScript");
// console.log(index);  // Output: 7

// 2 [includes()]
// let msg = "Welcome to my website";
// let result = msg.includes("website");
// console.log(result);  // Output: true

// 3 [startsWith() and endsWith()]
// let text = "Hello Hasnain";

// console.log(text.startsWith("Hello"));   // true
// console.log(text.endsWith("Hasnain"));   // true
// console.log(text.startsWith("H"));       // true
// console.log(text.endsWith("n"));         // true

// 4 [lastIndexOf()]
// let line = "Code JavaScript, learn JavaScript";

// let position = line.lastIndexOf("JavaScript");

// console.log(position);  // Output: 22


// ---------------------------------------

// Chapter # 24 Strings: Finding a character at a location

// ---------------------------------------


// 1 [charAt()]

// let userName = "Hasnain";

// let char = userName.charAt(3);

// console.log(char);  // Output: "n"


// 2 [Square brackets]

// let word = "JavaScript";

// console.log(word[4]);  // Output: "S"


// 3 [Last character]

// let city = "Karachi";

// let lastChar = city[city.length - 1];

// console.log(lastChar);  // Output: "i"


// ---------------------------------------

// Chapter # 25 Strings: Replacing characters

// ---------------------------------------
// 1 [one letter]

// let word = "banana";

// let newWord = word.replace("a", "o");

// console.log(newWord);  // Output: "bonana"


// 2 [one word]

// let sentence = "I love Python";

// let updated = sentence.replace("Python", "JavaScript");

// console.log(updated);  // Output: "I love JavaScript"


// 3 [replace all letter]

// let word = "banana";

// let replaced = word.replaceAll("a", "o");

// console.log(replaced);  // Output: "bonono"


// 4

// let sentence = "I like Python programming";


// let index = sentence.indexOf("Python");


// if (index !== -1) {

//   let updatedSentence = sentence.replace("Python", "JavaScript");

//   console.log("Updated:", updatedSentence);

// } else {

//   console.log("Word not found.");

// }
// --------------------------------------

// Chapter # 26 Rounding numbers

// --------------------------------------


// 1 [Math.round() : isma .5 ka bad one digit aga hojata han]

// let num = 4.6;

// let result = Math.round(num);

// console.log(result);  // Output: 5


// 2 [Math.floor() : isma point ka bad koy digit bhi ho voh osi digit pa hi raha ga]

// let num = 4.9;

// let result = Math.floor(num);

// console.log(result);  // Output: 4


// 3 [Math.ceil() : isma point ka bad koy digit bhi ho voh one digit aga chala jaye ga]

// let num = 4.1;

// let result = Math.ceil(num);

// console.log(result);  // Output: 5


// 4 [Math.random() : ya har bar new value create karta han]

// let num = Math.random();

// console.log(num);  // Example: 0.7325478


// 5 [.toFixed() : is ma one value deni hoti han jesa 2 value han to voh point ka bad 2 digit hi
lega]
```

```
// let price = 123.4567;
// let rounded = parseFloat(price.toFixed(2));
// console.log(rounded);  // Output: 123.46

// ---------------------------------------
// Chapter # 27 Generating random numbers
// ---------------------------------------

// 1 [Generating Random number]
// let num = Math.floor(Math.random() * 100) + 1;
// console.log(num);  // Output: 1 to 100

// 2 [Random number in custom range]
// function getRandom(min, max) {
//   return Math.floor(Math.random() * (max - min + 1)) + min;
// }
// console.log(getRandom(5, 20));  // Output: 5 to 20 ke darmiyan koi bhi number

// 3 [Random Dice Number (1 to 6)]
// let dice = Math.floor(Math.random() * 6) + 1;
// console.log("Dice rolled:", dice);  // Output: 1 se 6

// 4 [Random OTP Generator (4 digit)]
// let otp = Math.floor(1000 + Math.random() * 9000);
// console.log("Your OTP is:", otp);  // Output: e.g., 4832
// 5 [Random item from list (e.g., gift selector)]

// let gifts = ["Watch", "Book", "Perfume", "Bag", "Voucher"];

// let randomGift = gifts[Math.floor(Math.random() * gifts.length)];

// console.log("You won:", randomGift);


// ---------------------------------------

// Chapter # 28 Converting strings to integers and decimals

// ---------------------------------------


// 1 [Converting strings to integers]

// let str = "123"

// let convertInt = Number(str)

// console.log(typeof convertInt);


// 2 [Converting strings to decimals]

// let str = "12.34"

// let convertInt = Number(str)

// console.log(typeof convertInt);


// 3 [parseInt()]

// let str = "123";

// let num = parseInt(str);

// console.log(num);       // Output: 123

// console.log(typeof num); // Output: number


// 4 [parseFloat()]
// let price = "99.99";

// let actual = parseFloat(price);

// console.log(actual);        // Output: 99.99

// console.log(typeof actual); // Output: number


// 5 [Number()]

// let marks = "78.50";


// 6 [Invalid conversion]

// let wrong = "hello123";

// let result = parseInt(wrong);

// console.log(result);  // Output: NaN (Not a Number)


// 7 [difference String + Number]

// let a = "10";

// let b = 5;


// console.log(a + b); // Output: "105" → string + number = string

// console.log(Number(a) + b); // Output: 15 → dono number ban gaye


// ---------------------------------------

// Chapter # 29 Converting strings to numbers, numbers to strings

// ---------------------------------------


// 1 [Converting strings to numbers]

// let str1 = "123.45";
// let num1 = Number(str1);

// console.log(num1);        // Output: 123.45

// console.log(typeof num1);  // Output: number


// 2 [Converting numbers to strings]

// let num = 123;

// let str = num.toString();

// console.log(str);          // Output: "123"

// console.log(typeof str);    // Output: "string"


// --------------------------------------

// Chapter # 30 Controlling the length of decimals

// --------------------------------------


// 1 [.toFixed()]

// let price = 45.6789;

// let fixed = price.toFixed(2);

// console.log(fixed);         // Output: "45.68"

// console.log(typeof fixed);   // Output: "string"


// 2 [.toPrecision()]

// let num = 123.456;

// let result = num.toPrecision(4);

// console.log(result);  // Output: "123.5"


// 3 [.parseFloat()]
// let marks = 92.3786;

// let rounded = parseFloat(marks.toFixed(1));

// console.log(rounded);  // Output: 92.4


// --------------------------------------

// Chapter # 31 Getting the current date and time

// --------------------------------------


// 1 [Getting the current date]

// let newDate = new Date()

// let currDate = newDate.getDate()

// console.log(currDate);


// 2 [Getting the current time]

// let newDate = new Date()

// let currTime = newDate.getTime()

// console.log(currTime);


// --------------------------------------

// Chapter # 32 Extracting parts of the date and time

// --------------------------------------


// 1 [Extracting parts of the date]

// let newDate = new Date()


// let date  = newDate.getDate()
// let month = newDate.getMonth()
// let year  = newDate.getFullYear()

// let formattedDate = `Date:Month:Year = ${date}:${month}:${year}`
// console.log(formattedDate);

// 2 [Extracting parts of the time]
// let newDate = new Date()

// let hours   = newDate.getHours()
// let minutes = newDate.getMinutes()
// let seconds = newDate.getSeconds()

// let formattedTime = `hours:minutes:seconds = ${hours}:${minutes}:${seconds}`
// console.log(formattedTime);

// 3 [Combine formatted date & time]
// let d = new Date();

// let fullDate = d.getDate() + "/" + (d.getMonth()+1) + "/" + d.getFullYear();
// let fullTime = d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds();

// console.log("Today is:", fullDate);
// console.log("Current time:", fullTime);

// 4 [Show day name (custom)]
// let d = new Date();
// let days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

// let today = days[d.getDay()];
// console.log("Today is:", today);

// --------------------------------------
// Chapter # 33 [Specifying a date and time]
// --------------------------------------

// 1 [Using a Date String]
// let d = new Date("June 30, 2035");
// console.log(d);  // Output: Sat Jun 30 2035 ...

// 2 [Using year, month, day (number format)]
// let d = new Date(2035, 5, 30);  // Month is 0-based: 0 = Jan
// console.log(d);  // Output: Sat Jun 30 2035 ...

// 3 [Full date and time string]
// let d = new Date("June 30, 2035 15:30:00");
// console.log(d);  // Output: Sat Jun 30 2035 15:30:00 ...

// 4 [Specific time]
// let examDate = new Date(2025, 10, 15, 9, 30, 0);  // Nov 15, 2025 at 9:30 AM
// console.log("Exam starts on:", examDate);
// 5 [Comparing two dates]
// let today = new Date();
// let future = new Date("December 31, 2025");

// if (today < future) {
//   console.log("The future date is still to come.");
// } else {
//   console.log("The future date has passed.");
// }

// --------------------------------------
// Chapter # 34 [Changing elements of a date and time]
// --------------------------------------

// 1 [setFullYear()]
// let d = new Date();
// d.setFullYear(2006);
// console.log(d);  // Year is now 2006

// 2 [setMonth()]
// let d = new Date();
// d.setMonth(6);  // 6 means July
// console.log(d);  // Month is July

// 3 [setDate()]
// let d = new Date();
// d.setDate(6);  // 6th day of the month

// console.log(d);  // Day is 6


// 4 [setHours()]

// let d = new Date();

// d.setHours(6);  // 6 AM

// console.log(d);  // Time is 6:00 AM


// 5 [setMinutes()]

// let d = new Date();

// d.setMinutes(6);  // 6 minutes past the hour

// console.log(d);  // Time includes 6 minutes


// 6 [setSeconds()]

// let d = new Date();

// d.setSeconds(6);  // 6 seconds past the minute

// console.log(d);  // Time includes 6 seconds


// 7 [setMilliseconds()]

// let d = new Date();

// d.setMilliseconds(6);  // 6 milliseconds past the second

// console.log(d);  // Time includes 6 ms


// 8 [Task]


// // Step 1: Current date object
// let d = new Date();


// console.log("Original Date & Time:");
// console.log(d);


// // Step 2: Change all parts
// d.setFullYear(2030);      // Set year to 2030
// d.setMonth(6);            // Set month to July (6 = July)
// d.setDate(15);            // Set date to 15
// d.setHours(9);            // Set hour to 9 AM
// d.setMinutes(45);         // Set minutes to 45
// d.setSeconds(30);         // Set seconds to 30
// d.setMilliseconds(500);   // Set milliseconds to 500


// // Step 3: Show updated date and time
// console.log("\nUpdated Date & Time:");
// console.log("Date:", d.getDate() + "/" + (d.getMonth() + 1) + "/" + d.getFullYear());
// console.log("Time:", d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds() + "." +
d.getMilliseconds());


// ---------------------------------------
// Chapter # 35 [Functions]
// ---------------------------------------


// 1 [Function Syntax]
// function functionName() {
```

```
//   // code to run
// }

// 2 [Function Call]
// functionName();

// 3 [Basic Function]
// function sayHello() {
//   console.log("Hello, Hasnain!");
// }

// sayHello();  // Call the function

// 4 [Function with Parameters]
// function greet(name) {
//   console.log("Hello, " + name + "!");
// }

// greet("Hasnain");
// greet("Ali");

// 5 [Function with Return Value]
// function add(a, b) {
//   return a + b;
// }
// let result = add(5, 7);
// console.log("Result:", result);  // Output: 12

// 6 [Function Expression (variable me save karna)]
// let multiply = function(x, y) {
//   return x * y;
// }

// console.log(multiply(4, 3));  // Output: 12

// 7 [Arrow Function (short syntax)]
// const square = (n) => {
//   return n * n;
// };

// console.log(square(6));  // Output: 36

// 8 [Only Return]
// const double = n => n * 2;
// console.log(double(8));  // Output: 16

// --------------------------------------
// Chapter # 36 Functions: Passing them data
// --------------------------------------

// 1 [One parameter]
// function greet(name) {
//   console.log("Hello, " + name + "!");
// }

// greet("Hasnain");   // Output: Hello, Hasnain!

// 2 [Two parameters]
// function add(a, b) {
//   console.log("Sum is:", a + b);
// }

// add(5, 7);     // Output: Sum is: 12
// add(10, 20);   // Output: Sum is: 30

// 3 [Return with passed data]
// function multiply(x, y) {
//   return x * y;
// }

// let result = multiply(4, 3);
// console.log(result);  // Output: 12

// 4 [Passing strings]
// function welcome(message, name) {
//   console.log(message + ", " + name + "!");
// }
// welcome("Good morning", "Hasnain");  // Output: Good morning, Hasnain!


// 5 [Default value if data na mile]
// function sayHi(name = "Guest") {
//   console.log("Hi, " + name + "!");
// }


// sayHi("Hasnain");  // Output: Hi, Hasnain!
// sayHi();           // Output: Hi, Guest!


// 6 [Task : Grade Checker Function]
// function checkGrade(name, marks) {
//   let grade;


//   if (marks >= 80) {
//     grade = "A";
//   } else if (marks >= 70) {
//     grade = "B";
//   } else if (marks >= 60) {
//     grade = "C";
//   } else if (marks >= 50) {
//     grade = "D";
//   } else {
//     grade = "Fail";
//   }
//   console.log(name + "'s grade is: " + grade);
// }

// // Function calls with arguments
// checkGrade("Hasnain", 85);    // Output: Hasnain's grade is: A
// checkGrade("Ali", 67);        // Output: Ali's grade is: C
// checkGrade("Zara", 45);       // Output: Zara's grade is: Fail

// --------------------------------------
// Chapter # 37 Functions: Passing data back from them
// --------------------------------------

// 1 [Basic Syntax of Return]
// function functionName() {
//   return value;
// }

// 2 [Simple return value]
// function getPi() {
//   return 3.14;
// }

// let pi = getPi();   // function ne value wapas di
// console.log(pi);    // Output: 3.14
// 3 [Return after calculation]
// function square(num) {
//   return num * num;
// }

// let result = square(5);
// console.log("Square is:", result);  // Output: Square is: 25

// 4 [Function returns a message]
// function greet(name) {
//   return "Hello, " + name + "!";
// }

// let msg = greet("Hasnain");
// console.log(msg);   // Output: Hello, Hasnain!

// 5 [Return with if/else logic]
// function checkPass(marks) {
//   if (marks >= 50) {
//     return "Pass";
//   } else {
//     return "Fail";
//   }
// }

// let result = checkPass(45);
// console.log("Result:", result);  // Output: Result: Fail


// 6 [Task : Bill Calculator]
// function calculateBill(price, taxRate) {
//   let total = price + (price * taxRate / 100);
//   return total.toFixed(2);  // returns string like "107.50"
// }


// let bill = calculateBill(100, 7.5);
// console.log("Total Bill:", bill);  // Output: Total Bill: 107.50


// --------------------------------------
// Chapter # 38 Functions: Local vs. global variables
// --------------------------------------


// 1 [Global Variable Example]
// let username = "Hasnain";  // Global variable


// function greet() {
//   console.log("Hello, " + username + "!");
// }


// greet();  // Output: Hello, Hasnain!


// 2 [Local Variable Example]
// function showMessage() {
```

```
//   let message = "Welcome!";
//   console.log(message);
// }

// showMessage();       // Output: Welcome!
// console.log(message); // ❌ Error: message is not defined

// 3 [Both Together Example]
// let appName = "MyApp";  // Global

// function printAppInfo() {
//   let version = "1.0";  // Local
//   console.log(appName + " v" + version);
// }

// printAppInfo();         // Output: MyApp v1.0
// console.log(appName);   // ✅ OK
// console.log(version);   // ❌ Error: version is not defined

// 4 [Variable Shadowing (Local overrides Global)]
// let score = 100;  // Global

// function updateScore() {
//   let score = 50;  // Local (same name!)
//   console.log("Inside:", score);
// }
// updateScore();      // Output: Inside: 50
// console.log("Global:", score);  // Output: Global: 100


// --------------------------------------
// Chapter # 39 switch statements: How to start them
// --------------------------------------


// 1 [Basic Syntax of switch]
// switch (expression) {
//   case value1:
//     // code block
//     break;
//   case value2:
//     // code block
//     break;
//   default:
//     // code block
// }


// 2 [Weekday Checker]
// let day = "Monday";


// switch (day) {
//   case "Monday":
//     console.log("Start of the week!");
//     break;

//   case "Friday":
//     console.log("Weekend is near!");
//     break;

//   case "Sunday":
//     console.log("Relax, it's Sunday!");
//     break;

//   default:
//     console.log("Normal day.");
// }

// 3 [Grading System]
// let grade = "B";

// switch (grade) {
//   case "A":
//     console.log("Excellent!");
//     break;
//   case "B":
//     console.log("Good job!");
//     break;
//   case "C":
//     console.log("You passed.");
//     break;
//   case "D":
//     console.log("Try harder next time.");
//     break;
//   default:
//     console.log("Invalid grade");
// }

// 4 [Without break (wrong result)]
// let color = "red";

// switch (color) {
//   case "red":
//     console.log("Stop!");
//   case "green":
//     console.log("Go!");
//   default:
//     console.log("Unknown color");
// }

// // Output:
// // Stop!
// // Go!
// // Unknown color

// --------------------------------------
// Chapter # 40 switch statements: How to complete them
// ---------------------------------------

// 1 [Structure]
// switch (expression) {
//   case value1:
//     // run this code
//     break;

//   case value2:
//     // run this code
//     break;

//   // more cases...

//   default:
//     // run this if nothing matches
// }

// 2 [Day of the Week Checker]
// let day = "Tuesday";

// switch (day) {
//   case "Monday":
//     console.log("Start of the week");
//     break;
//   case "Tuesday":
//     console.log("Still early in the week");
//     break;

//   case "Wednesday":
//     console.log("Midweek");
//     break;

//   case "Thursday":
//     console.log("Almost there");
//     break;

//   case "Friday":
//     console.log("Weekend is coming");
//     break;

//   case "Saturday":
//   case "Sunday":
//     console.log("Weekend!");
//     break;

//   default:
//     console.log("Invalid day");
// }
// 3 [Grouped Cases]
// let signal = "red";

// switch (signal) {
//   case "red":
//     console.log("Stop");
//     break;

//   case "yellow":
//     console.log("Get ready");
//     break;

//   case "green":
//     console.log("Go");
//     break;

//   default:
//     console.log("Signal malfunction");
// }

// 4 [Simple Calculator]
// function calculate(num1, num2, operator) {
//   let result;

//   switch (operator) {
//     case "+":
```

```
//      result = num1 + num2;
//      break;

//    case "-":
//      result = num1 - num2;
//      break;

//    case "*":
//      result = num1 * num2;
//      break;

//    case "/":
//      if (num2 === 0) {
//        result = "Error: Divide by 0";
//      } else {
//        result = num1 / num2;
//      }
//      break;

//    default:
//      result = "Invalid operator";
//  }

//  console.log("Result:", result);
// }
// // ✅ Function calls:

// calculate(10, 5, "+");  // Output: Result: 15

// calculate(8, 2, "*");   // Output: Result: 16

// calculate(9, 0, "/");   // Output: Result: Error: Divide by 0

// calculate(6, 3, "-");   // Output: Result: 3

// calculate(4, 2, "%");   // Output: Invalid operator


// --------------------------------------

// Chapter # 41 while loops

// --------------------------------------


// 1 [Syntax]

// while (condition) {

//   // code to run again and again

// }


// 2 [Count 1 to 5]

// let i = 1;


// while (i <= 5) {

//   console.log("Number:", i);

//   i++;

// }


// // Output:

// // Number: 1
// // Number: 2

// // ...

// // Number: 5


// 3 [Print even numbers (2 to 10)]

// let num = 2;


// while (num <= 10) {

//   console.log(num);

//   num += 2;

// }


// // Output: 2 4 6 8 10


// 4 [User guess game (simple simulation)]

// let guess = 3;

// let input = 0;


// while (input !== guess) {

//   input++;  // Simulate user guessing

//   console.log("Guessing:", input);

// }


// console.log("Correct guess!");


// 5 [Infinite Loop]
// let i = 1;

// while (i <= 5) {
//   // Missing i++ will cause infinite loop!
//   console.log(i);
// }

// ---------------------------------------
// Chapter # 42 do...while loops
// ---------------------------------------

// 1 [Syntax]
// do {
//   // code to run at least once
// } while (condition);

// 2 [Print 1 to 5]
// let i = 1;

// do {
//   console.log("Number:", i);
//   i++;
// } while (i <= 5);

// // Output:
// // Number: 1
// // Number: 2

// // ...

// // Number: 5


// 3 [Run once even if condition is false]

// let x = 10;


// do {

//   console.log("This will run even though x > 5");

// } while (x < 5);


// // Output:

// // This will run even though x > 5


// 4 [User Login Attempts]

// let password = "1234";

// let userInput;

// let attempts = 0;


// do {

//   userInput = "wrong" + attempts; // simulate wrong inputs

//   console.log("Trying password:", userInput);

//   attempts++;

// } while (userInput !== password && attempts < 3);


// if (userInput === password) {
```

```
//   console.log("Login successful");
// } else {
//   console.log("Login failed after 3 tries");
// }

// 5 [Math Table of a Number]
// let number = 5;
// let i = 1;

// do {
//   console.log(`${number} x ${i} = ${number * i}`);
//   i++;
// } while (i <= 10);

// 6 [Ask for Correct PIN (Max 4 Attempts)]
// let correctPIN = "4321";
// let enteredPIN = "";
// let tries = 0;

// do {
//   enteredPIN = "0000"; // simulate wrong PIN
//   console.log("Try:", tries + 1);
//   tries++;
// } while (enteredPIN !== correctPIN && tries < 4);

// if (enteredPIN === correctPIN) {
```

```
//   console.log("Access Granted");

// } else {

//   console.log("Card Blocked after 4 wrong attempts");

// }


// --------------------------------------

// Chapter # 43 Placing scripts

// --------------------------------------


// 1 [Inline Script (Inside HTML tag)]

// <button onclick="alert('Hello, Hasnain!')">Click Me</button>


// 2 [Internal Script (inside <script> tag in HTML)]

// <script>

//   console.log("Hello from script!");

//   alert("Welcome, Hasnain!");

// </script>


// 3 [External Script File (.js file)]

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


// --------------------------------------

// Chapter # 44 Commenting

// --------------------------------------


// 1 [Single-line Comment]
// let name = "Hasnain"; // Storing user's name

// 2 [Multi-line Comment]
/*
  This is a multi-line comment.
  You can write as many lines as you want.
  Good for long explanations.
*/
// let age = 25;

// 3 [Tip: Tools help too]
// Ctrl + / → quickly comment/uncomment line
// Multi-line comment auto complete hota hai

// --------------------------------------
// Chapter # 45 Events: link
// --------------------------------------

// 1 [Simple alert on link click]
// <a href="#" onClick="alert('Hi');">Click</a>

// 2 [Prevent link from actually going to another page]
// <a href="https://google.com" onclick="alert('You clicked Google!'); return false;">Go to Google</a>

// 3 [function call from link]
// <a href="#" onclick="showMessage()">Click to see message</a>


// <script>
//   function showMessage() {
//     alert("Welcome, Hasnain bhai!");
//   }
// </script>


// 4 [Modern Way (Event Listener)]
// <a href="#" id="myLink">Click Here</a>


// document.getElementById("myLink").addEventListener("click", function(e) {
//   e.preventDefault();  // prevent link behavior
//   alert("Modern JS method triggered!");
// });


// 5 [href="#"]
// Jab link ka actual page navigation required nahi ho
// Jab aap sirf JavaScript se koi action chalwana chahte ho


// --------------------------------------
// Chapter # 46 Events: button
// --------------------------------------


// 1 [Basic Button Event (Inline)]
// <button onclick="alert('Button clicked!')">Click Me</button>
// 2 [Best Practice: Button with External JavaScript Function]
// <!-- HTML -->
// <button onclick="sayHello()">Say Hello</button>

// <script>
//   function sayHello() {
//     alert("Hello, Hasnain bhai!");
//   }
// </script>

// 3 [Modern Way: Event Listener]
// <button id="myBtn">Click Me</button>

// <script>
//   document.getElementById("myBtn").addEventListener("click", function() {
//     alert("Modern method: Button clicked!");
//   });
// </script>

// 4 [Change text on button click]
// <p id="message">Original Text</p>
// <button onclick="document.getElementById('message').innerText = 'Text Changed!'">Change
Text</button>

// 5 [Show/hide element]
// <button onclick="toggleBox()">Toggle Box</button>

// <div id="box" style="display: none; background: #eee; padding: 10px; margin-top:
10px;">This is a box</div>


// <script>

//   function toggleBox() {

//     const box = document.getElementById("box");

//     box.style.display = box.style.display === "none" ? "block" : "none";

//   }

// </script>


// 6 [Change background color]

// <button onclick="document.body.style.backgroundColor = 'lightblue'">Change
Background</button>


// 7 [Counter Button]

// <p>Clicked <span id="count">0</span> times</p>

// <button onclick="increase()">Click Me</button>


// <script>

//   let counter = 0;


//   function increase() {

//     counter++;

//     document.getElementById("count").innerText = counter;

//   }

// </script>
// --------------------------------------

// Chapter # 47 Events: mouse

// --------------------------------------


// 1 [onmouseover & onmouseout]

/*<div onmouseover="this.style.background='lightgreen'"

  onmouseout="this.style.background='white'"

  style="width: 200px; height: 100px; border: 1px solid #aaa;">

  Hover over me!

</div>*/


// 2 [onclick + ondblclick]

/*<p onclick="alert('Single Click!')" ondblclick="alert('Double Click!')">

  Click or Double Click this text

</p>*/


// 3 [onmousemove - Show X,Y Position]

/*<div onmousemove="showCoords(event)"

     style="height: 150px; border: 1px solid black;">

  Move your mouse here

</div>


<p id="coords"></p>


<script>
```

```
  function showCoords(e) {
    document.getElementById("coords").innerText =
      "X: " + e.clientX + " | Y: " + e.clientY;
  }
</script>*/

// 4 [onmousedown / onmouseup color change]
/*<div onmousedown="this.style.background='orange'"
     onmouseup="this.style.background='lightgray'"
     style="width: 150px; height: 100px; border: 1px solid #aaa;">
  Hold and Release
</div>*/

// 5 [Hover to Show Hidden Text (onmouseover / onmouseout)]
/*<div onmouseover="document.getElementById('hidden').style.display = 'block'"
     onmouseout="document.getElementById('hidden').style.display = 'none'"
     style="width: 200px; padding: 10px; border: 1px solid gray;">
  Hover here to reveal secret
</div>

<p id="hidden" style="display: none; color: red;">🎉 Surprise! You found it!</p>*/

// 6 [Image Change on Hover (onmouseover)]
/*<img src="https://via.placeholder.com/150"
  onmouseover="this.src='https://via.placeholder.com/150/ff0000'"
  onmouseout="this.src='https://via.placeholder.com/150'" />*/
```

```
// 7 [Draw with Mouse (onmousemove)]

/*<div onmousemove="draw(event)"

     style="width: 300px; height: 150px; border: 1px solid black; position: relative;"
id="canvas"></div>


<script>

  function draw(e) {

    const dot = document.createElement("div");

    dot.style.width = dot.style.height = "5px";

    dot.style.background = "blue";

    dot.style.position = "absolute";

    dot.style.left = e.offsetX + "px";

    dot.style.top = e.offsetY + "px";

    e.target.appendChild(dot);

  }

</script>*/


// 8 [Toggle Background Color (onclick)]

/*<div id="colorBox" onclick="toggleColor()"

     style="width: 200px; height: 100px; background: lightgray;"></div>


<script>

  let isGray = true;


  function toggleColor() {
    const box = document.getElementById("colorBox");
    box.style.background = isGray ? "lightgreen" : "lightgray";
    isGray = !isGray;
  }
</script>*/

// 9 [Count Mouse Clicks]
/*<p>You clicked <span id="clickCount">0</span> times!</p>
<div onclick="countClick()"
     style="width: 200px; height: 80px; border: 1px solid blue;"></div>

<script>
  let clicks = 0;
  function countClick() {
    clicks++;
    document.getElementById("clickCount").innerText = clicks;
  }
</script>*/

// 10 [Mouse Trail Effect (mouse jahan jaye, circle wahan aaye)]
/*<!DOCTYPE html>
<html>
<head>
  <title>Mouse Trail Effect</title>
  <style>
    body {
      height: 100vh;
      margin: 0;
      overflow: hidden;
    }

    #circle {
      width: 20px;
      height: 20px;
      background: red;
      border-radius: 50%;
      position: absolute;
      pointer-events: none; // prevent blocking mouse events
      transition: all 0.05s linear;
    }
  </style>
</head>
<body>

  <div id="circle"></div>

  <script>
    const circle = document.getElementById("circle");

    document.addEventListener("mousemove", function (e) {
      circle.style.left = e.pageX + "px";
      circle.style.top = e.pageY + "px";
    });
  </script>

</body>
</html>*/

// ---------------------------------------
// Chapter # 48 Events: fields
// ---------------------------------------

// 1 [onfocus and onblur - Highlight input]
/* <input type="text" onfocus="this.style.background='lightyellow'"
      onblur="this.style.background='white'" placeholder="Enter your name" /> */

// 2 [onchange - Dropdown Selection]
/* <select onchange="alert('You selected: ' + this.value)">
  <option>Select city</option>
  <option>Karachi</option>
  <option>Lahore</option>
  <option>Islamabad</option>
</select> */

// 3 [oninput - Live typing preview]
/* <input type="text" oninput="document.getElementById('preview').innerText = this.value"
       placeholder="Type something..." />
```

```
<p>Live preview: <span id="preview"></span></p> */


// 4 [onkeyup - Count characters]

/*<input type="text" id="message" onkeyup="countChars()" placeholder="Type your message"
/>

<p>Characters: <span id="charCount">0</span></p>


<script>

  function countChars() {

    let text = document.getElementById("message").value;

    document.getElementById("charCount").innerText = text.length;

  }

</script>*/


// 5 [onblur for Validation]

/*<input type="email" id="email" onblur="validateEmail()" placeholder="Enter email" />

<p id="error" style="color: red;"></p>


<script>

  function validateEmail() {

    let email = document.getElementById("email").value;

    let error = document.getElementById("error");


    if (!email.includes("@")) {

      error.innerText = "❌ Invalid email";

    } else {
      error.innerText = "";
    }
  }
</script>*/

// ---------------------------------------
// Chapter # 49 Reading field values
// ---------------------------------------

//  1 [Syntax]
// let value = document.getElementById("fieldID").value;

//  2 [Text Field Value Read karna]
/* <input type="text" id="username" placeholder="Enter your name" />
<button onclick="getName()">Show Name</button>

<script>
  function getName() {
    let name = document.getElementById("username").value;
    alert("Your name is: " + name);
  }
</script> */

//  3 [Textarea ka Value Read karna]
/* <textarea id="msg" placeholder="Type message here..."></textarea>
<button onclick="readMessage()">Read</button>
```

```
<script>
  function readMessage() {
    let message = document.getElementById("msg").value;
    console.log("Message is:", message);
  }
</script> */

// 4 [Checkbox ka Value aur Status]
/* <input type="checkbox" id="agree" /> I agree
<button onclick="checkAgreement()">Submit</button>

<script>
  function checkAgreement() {
    let isChecked = document.getElementById("agree").checked;
    alert(isChecked ? "You agreed!" : "Please agree first.");
  }
</script> */

// 5 [Radio Button Value Read karna]
/* <p>Select gender:</p>
<input type="radio" name="gender" value="Male" /> Male
<input type="radio" name="gender" value="Female" /> Female
<button onclick="readGender()">Check</button>

<script>
```

```
  function readGender() {
    let selected = document.querySelector('input[name="gender"]:checked');
    alert("Selected gender: " + (selected ? selected.value : "None"));
  }
</script> */

// 6 [Dropdown (Select) Value Read karna]
/* <select id="city">
  <option>Karachi</option>
  <option>Lahore</option>
  <option>Islamabad</option>
</select>
<button onclick="getCity()">Get City</button>

<script>
  function getCity() {
    let city = document.getElementById("city").value;
    alert("Selected city: " + city);
  }
</script> */

// --------------------------------------
// Chapter # 50 Setting field values
// --------------------------------------

// 1 [Basic Syntax]
// document.getElementById("fieldID").value = "Your Value";


// 2 [Set Text Value]
/* <input type="text" id="username" />
<button onclick="setName()">Set Name</button>


<script>
  function setName() {
    document.getElementById("username").value = "Hasnain";
  }
</script> */


// 3 [Set Textarea Value]
/* <textarea id="message" rows="3" cols="30"></textarea>
<button onclick="setMessage()">Set Message</button>


<script>
  function setMessage() {
    document.getElementById("message").value = "Hello, welcome to JavaScript class!";
  }
</script> */


// 4 [Set Checkbox Checked/Unchecked]
/* <input type="checkbox" id="subscribe" /> Subscribe me
<button onclick="checkBox()">Auto Check</button>
```

```
<script>
  function checkBox() {
    document.getElementById("subscribe").checked = true;
  }
</script> */

// 5 [Set Selected Radio Button]
/* <p>Select gender:</p>
<input type="radio" name="gender" value="Male" id="male" /> Male
<input type="radio" name="gender" value="Female" id="female" /> Female
<br>
<button onclick="selectGender()">Auto Select Female</button>

<script>
  function selectGender() {
    document.getElementById("female").checked = true;
  }
</script> */

// 6 [Set Dropdown Value]
/* <select id="city">
  <option value="Karachi">Karachi</option>
  <option value="Lahore">Lahore</option>
  <option value="Islamabad">Islamabad</option>
</select>
```

```
<button onclick="setCity()">Select Lahore</button>


<script>
  function setCity() {
    document.getElementById("city").value = "Lahore";
  }
</script> */


// ---------------------------------------
// Chapter # 51 Reading and setting paragraph text
// ---------------------------------------


// 1 [Reading Paragraph Text]
// let text = document.getElementById("paraID").innerText;


// 2 [Setting Paragraph Text]
// document.getElementById("paraID").innerText = "New content";


// 3 [Read paragraph and show alert]
/* <p id="para1">Welcome to JavaScript Class!</p>
<button onclick="readText()">Read Text</button>


<script>
  function readText() {
    let text = document.getElementById("para1").innerText;
    alert("Paragraph says: " + text);
  }
</script> */


// 4 [Set paragraph text on button click]
/* <p id="para2">This is old text.</p>
<button onclick="changeText()">Change Text</button>


<script>
  function changeText() {
    document.getElementById("para2").innerText = "Now the text is updated!";
  }
</script> */


// 5 [Set paragraph text from input field]
/* <input type="text" id="inputBox" placeholder="Type something..." />
<button onclick="updatePara()">Show in Paragraph</button>


<p id="output"></p>


<script>
  function updatePara() {
    let userText = document.getElementById("inputBox").value;
    document.getElementById("output").innerText = userText;
  }
</script> */
```

```
// 6 [Clear paragraph text]

/* <p id="info">This will be removed when you click the button.</p>

<button onclick="clearText()">Clear</button>


<script>

  function clearText() {

    document.getElementById("info").innerText = "";

  }

</script> */


// 7 [Count number of characters in a paragraph]

/* <p id="longText">JavaScript is a powerful language used for web development.</p>

<button onclick="countChars()">Count Characters</button>

<p id="countResult"></p>


<script>

  function countChars() {

    let text = document.getElementById("longText").innerText;

    document.getElementById("countResult").innerText = "Total characters: " + text.length;

  }

</script> */


// 8 [Task]

/* <p id="myPara">this is my original paragraph.</p>

<button onclick="beautify()">Make Beautiful</button> */
```

```
// function beautify() {

//   document.getElementById("myPara").innerHTML =

//     "This is my <strong><span style='color: green;'>updated</span></strong> paragraph with "
+

//     "<em><span style='color: blue;'>style</span></em> and <u>formatting</u>!";

// }


// --------------------------------------

// Chapter # 52 Manipulating images and text

// --------------------------------------


// 1 [Change Image on Button Click]

/* <img id="myImg" src="https://via.placeholder.com/150" />

<br>

<button onclick="changeImage()">Change Image</button>


<script>

  function changeImage() {

    document.getElementById("myImg").src = "https://via.placeholder.com/150/ff0000";

  }

</script> */


// 2 [Show/Hide Image]

/* <img id="pic" src="https://via.placeholder.com/120" />

<br>

<button onclick="toggleImage()">Toggle Image</button>
```

```
<script>
  function toggleImage() {
    let img = document.getElementById("pic");
    img.style.display = img.style.display === "none" ? "block" : "none";
  }
</script> */

// 3 [Resize Image Dynamically]
/* <img id="resImg" src="https://via.placeholder.com/100" />
<br>
<button onclick="resize()">Make Bigger</button>

<script>
  function resize() {
    document.getElementById("resImg").style.width = "200px";
  }
</script> */

// 4 [Image changes automatically every 2 seconds (slideshow)]
/* <img id="slideImg" src="https://via.placeholder.com/200/0000ff" />
<script>
  const images = [
    "https://via.placeholder.com/200/0000ff",
    "https://via.placeholder.com/200/ff0000",
    "https://via.placeholder.com/200/00ff00",
    "https://via.placeholder.com/200/f0f0f0"
  ];

  let index = 0;

  setInterval(() => {
    index = (index + 1) % images.length;
    document.getElementById("slideImg").src = images[index];
  }, 2000); // every 2 seconds
</script> */

// 5 [Manipulating Text]
/* <p id="text">This is the original text.</p>
<button onclick="changeText()">Change Text</button>

<script>
  function changeText() {
    document.getElementById("text").innerText = "🎉 Now the text is updated!";
  }
</script> */

// 6 [Change Text Style Dynamically]
/* <p id="fancyText">Make me fancy!</p>
<button onclick="styleText()">Style Text</button>

<script>
```

```
  function styleText() {
    let t = document.getElementById("fancyText");
    t.style.color = "blue";
    t.style.fontSize = "24px";
    t.style.fontWeight = "bold";
    t.style.fontFamily = "Arial";
  }
</script> */

// 7 [Rotate Text or Image (CSS transform)]
/* <img id="rotImg" src="https://via.placeholder.com/100" />
<br>
<button onclick="rotateImg()">Rotate</button>

<script>
  function rotateImg() {
    document.getElementById("rotImg").style.transform = "rotate(180deg)";
  }
</script> */

// 8 [Combine: Image + Text Change Together]
/* <img id="comboImg" src="https://via.placeholder.com/100" />
<p id="comboText">Old text here...</p>
<br>
<button onclick="updateBoth()">Change Both</button>
```

```
<script>
  function updateBoth() {
    document.getElementById("comboImg").src = "https://via.placeholder.com/100/0000ff";
    document.getElementById("comboText").innerText = "🎉 New image & text loaded!";
  }
</script> */

// 9 [Typing Effect (1 letter at a time)]
/* <p id="typeText"></p>
<script>
  const message = "Welcome, Hasnain bhai! 😄 This text is typing...";
  let i = 0;

  function typeWriter() {
    if (i < message.length) {
      document.getElementById("typeText").innerText += message.charAt(i);
      i++;
      setTimeout(typeWriter, 100); // delay between letters
    }
  }

  typeWriter();
</script> */

// ---------------------------------------
// Chapter # 53 Swapping images
// ---------------------------------------

// 1 [Swap on Button Click]
/* <img id="swapImg" src="https://via.placeholder.com/150/00f" />
<br>
<button onclick="swap()">Swap Image</button>

<script>
  let swapped = false;

  function swap() {
    const img = document.getElementById("swapImg");
    if (swapped) {
      img.src = "https://via.placeholder.com/150/00f"; // Original
    } else {
      img.src = "https://via.placeholder.com/150/f00"; // New
    }
    swapped = !swapped;
  }
</script> */

// 2 [Swap on Mouse Hover]
/* <img id="hoverImg"
     src="https://via.placeholder.com/200/000"
     onmouseover="this.src='https://via.placeholder.com/200/green'"
     onmouseout="this.src='https://via.placeholder.com/200/000'" /> */
```

```
// 3 [Swap Between 3 Images (Like Gallery)]
/* <img id="gallery" src="https://via.placeholder.com/200/0000ff" />
<br>
<button onclick="nextImage()">Next</button>

<script>
  const imgs = [
    "https://via.placeholder.com/200/0000ff",
    "https://via.placeholder.com/200/ff0000",
    "https://via.placeholder.com/200/00ff00"
  ];
  let current = 0;

  function nextImage() {
    current = (current + 1) % imgs.length;
    document.getElementById("gallery").src = imgs[current];
  }
</script> */

// 4 [Swap Image Based on Input Field]
/* <input type="text" id="imgName" placeholder="Type red / blue / green" />
<button onclick="swapByInput()">Swap Image</button>
<br><br>
<img id="imgBox" src="https://via.placeholder.com/150" />
```

```
}

<script>
  function swapByInput() {
    let val = document.getElementById("imgName").value.toLowerCase();
    let img = document.getElementById("imgBox");

    if (val === "red") img.src = "https://via.placeholder.com/150/ff0000";
    else if (val === "blue") img.src = "https://via.placeholder.com/150/0000ff";
    else if (val === "green") img.src = "https://via.placeholder.com/150/00ff00";
    else img.src = "https://via.placeholder.com/150"; // default
  }
</script> */

// 5 [Swap Image on Double Click]
/* <img id="dblImg" src="https://via.placeholder.com/150/888" ondblclick="swapDouble()" />

<script>
  function swapDouble() {
    document.getElementById("dblImg").src = "https://via.placeholder.com/150/f90";
  }
</script> */

// 6 [Image Swap with Animation (CSS + JS)]
/* <style>
  #fadeImg {
    transition: all 0.5s ease;
  }
```

```
</style>

<img id="fadeImg" src="https://via.placeholder.com/200/000" />
<br>
<button onclick="fadeSwap()">Fade Swap</button>

<script>
  function fadeSwap() {
    const img = document.getElementById("fadeImg");
    img.style.opacity = 0;
    setTimeout(() => {
      img.src = "https://via.placeholder.com/200/f00";
      img.style.opacity = 1;
    }, 500);
  }
</script> */

// 7 [Image Swap on Random Button Press]
/* <img id="randomImg" src="https://via.placeholder.com/150/999" />
<br>
<button onclick="randomSwap()">Random Image</button>

<script>
  function randomSwap() {
    const colors = ["ff0000", "00ff00", "0000ff", "ffff00", "999999"];
    const random = colors[Math.floor(Math.random() * colors.length)];
    document.getElementById("randomImg").src = `https://via.placeholder.com/150/${random}`;
  }
</script> */

// --------------------------------------
// Chapter # 54 Swapping images and setting classes
// --------------------------------------

// 1 [Swap Image & Toggle Class on Click]
/* <style>
  .redBorder {
    border: 5px solid red;
    border-radius: 10px;
  }

  .blueBorder {
    border: 5px solid blue;
    border-radius: 10px;
  }
</style>

<img id="myPic" src="https://via.placeholder.com/200/00f" class="blueBorder" />
<br>
<button onclick="swapImageAndClass()">Swap</button>

<script>
```

```
  let isBlue = true;

  function swapImageAndClass() {
    const img = document.getElementById("myPic");

    if (isBlue) {
      img.src = "https://via.placeholder.com/200/f00";
      img.className = "redBorder";
    } else {
      img.src = "https://via.placeholder.com/200/00f";
      img.className = "blueBorder";
    }

    isBlue = !isBlue;
  }
</script> */

// 2 [classList.add, remove, toggle]
/* <style>
  .shadow {
    box-shadow: 0 0 10px black;
  }

  .rotate {
    transform: rotate(10deg);
    transition: 0.3s;
  }
</style>

<img id="effectImg" src="https://via.placeholder.com/150" />
<br>
<button onclick="applyEffects()">Apply Effects</button>

<script>
  function applyEffects() {
    const img = document.getElementById("effectImg");

    img.classList.toggle("shadow");
    img.classList.toggle("rotate");

    img.src = img.src.includes("150")
      ? "https://via.placeholder.com/150/0f0"
      : "https://via.placeholder.com/150";
  }
</script> */

// 3 [Add/Remove Class Without Changing Other Classes]
// // Add a class
// element.classList.add("myClass");

// // Remove a class
// element.classList.remove("myClass");
// // Toggle a class (add if not present, remove if already there)
// element.classList.toggle("myClass");


// // Check if a class exists
// element.classList.contains("myClass");


// 4 [Swap Image Every Time Button Clicks]
/* <img id="photo" src="https://via.placeholder.com/150/0000FF" />
<br>
<button onclick="change()">Change Image</button>


<script>
  let isOriginal = true;


  function change() {
    const img = document.getElementById("photo");
    if (isOriginal) {
      img.src = "https://via.placeholder.com/150/FF0000";
    } else {
      img.src = "https://via.placeholder.com/150/0000FF";
    }
    isOriginal = !isOriginal;
  }
</script> */
```

```
// 5 [Change Image Based on User Input]

/* <input id="color" placeholder="red / green / blue" />

<button onclick="updateImage()">Show Image</button>

<br><br>

<img id="colorImg" src="https://via.placeholder.com/150" />


<script>

  function updateImage() {

    const color = document.getElementById("color").value.toLowerCase();

    const img = document.getElementById("colorImg");


    if (color === "red") img.src = "https://via.placeholder.com/150/ff0000";

    else if (color === "green") img.src = "https://via.placeholder.com/150/00ff00";

    else if (color === "blue") img.src = "https://via.placeholder.com/150/0000ff";

    else img.src = "https://via.placeholder.com/150";

  }

</script> */


// 6 [Auto Change Image Every 3 Seconds (Slideshow)]

/* <img id="autoImg" src="https://via.placeholder.com/150/000" />

<script>

  const imgList = [

    "https://via.placeholder.com/150/000",

    "https://via.placeholder.com/150/f00",

    "https://via.placeholder.com/150/0f0",

    "https://via.placeholder.com/150/00f"
  ];
  let index = 0;

  setInterval(() => {
    index = (index + 1) % imgList.length;
    document.getElementById("autoImg").src = imgList[index];
  }, 3000);
</script> */

// 7 [Change Image Randomly on Button Click]
/* <img id="randomImage" src="https://via.placeholder.com/150" />
<br>
<button onclick="randomPic()">Random Image</button>

<script>
  function randomPic() {
    const colors = ["ff0000", "00ff00", "0000ff", "ffff00", "ff00ff"];
    const color = colors[Math.floor(Math.random() * colors.length)];
    document.getElementById("randomImage").src = `https://via.placeholder.com/150/${color}`;
  }
</script> */

// 8 [Change Image Based on Time of Day]
/* <img id="timeImage" src="" />

<script>
```

```
  const hour = new Date().getHours();

  const img = document.getElementById("timeImage");


  if (hour < 12) {

    img.src = "https://via.placeholder.com/150/ffffcc"; // Morning

  } else if (hour < 18) {

    img.src = "https://via.placeholder.com/150/ffd700"; // Afternoon

  } else {

    img.src = "https://via.placeholder.com/150/333333"; // Night

  }

</script> */


// --------------------------------------

// Chapter # 55 Setting styles

// --------------------------------------


// 1 [Syntax]

// document.getElementById("elementID").style.property = "value";


// 2 [Change Text Color and Size]

/* <p id="myText">Change my style!</p>

<button onclick="changeStyle()">Style Me</button>


<script>

  function changeStyle() {

    const text = document.getElementById("myText");
    text.style.color = "blue";

    text.style.fontSize = "24px";

    text.style.fontWeight = "bold";

    text.style.fontFamily = "Arial";

  }

</script> */


// 3 [Style a Box]

/* <div id="box" style="width: 100px; height: 100px; background: grey;"></div>

<br>

<button onclick="styleBox()">Style the Box</button>


<script>

  function styleBox() {

    const box = document.getElementById("box");

    box.style.backgroundColor = "green";

    box.style.borderRadius = "10px";

    box.style.boxShadow = "0 0 10px black";

    box.style.transform = "rotate(10deg)";

  }

</script> */


// 4 [Hide/Show Element with Styling]

/* <p id="para">You can hide me!</p>

<button onclick="togglePara()">Toggle Visibility</button>
```

```
<script>
  function togglePara() {
    const p = document.getElementById("para");
    if (p.style.display === "none") {
      p.style.display = "block";
    } else {
      p.style.display = "none";
    }
  }
</script> */

// 5 [Add CSS Class Instead (Clean Way)]
/* <style>
  .fancy {
    color: purple;
    font-size: 28px;
    text-shadow: 1px 1px 2px gray;
  }
</style>

<p id="fancyText">Make me fancy!</p>
<button onclick="addClass()">Add Class</button>

<script>
  function addClass() {
    document.getElementById("fancyText").classList.add("fancy");
  }
</script> */

// 6 [Dark Mode Toggle]
/* <body id="body" style="background-color: white; color: black;">
  <h2>Welcome to my page</h2>
  <button onclick="toggleMode()">Toggle Dark Mode</button>

  <script>
    function toggleMode() {
      const body = document.getElementById("body");
      if (body.style.backgroundColor === "white") {
        body.style.backgroundColor = "black";
        body.style.color = "white";
      } else {
        body.style.backgroundColor = "white";
        body.style.color = "black";
      }
    }
  </script>
</body> */

// 7 [Live Style Update Based on Input]
/* <input id="colorInput" placeholder="Enter color like red, blue, green" />
<button onclick="changeColor()">Change Background</button>
<br><br>
```

```
<div id="colorBox" style="width:150px; height:150px; background-color:lightgray;"></div>


<script>
  function changeColor() {
    const input = document.getElementById("colorInput").value;
    document.getElementById("colorBox").style.backgroundColor = input;
  }
</script> */


// 8 [Animate with Style (on click)]
/* <div id="anim" style="width:100px; height:100px; background:orange;"></div>
<br>
<button onclick="animateBox()">Animate</button>


<script>
  function animateBox() {
    const box = document.getElementById("anim");
    box.style.transition = "all 0.5s ease";
    box.style.transform = "translateX(200px) rotate(15deg)";
    box.style.backgroundColor = "tomato";
    box.style.borderRadius = "50%";
  }
</script> */


// 9 [Font Style Picker]
/* <select id="fontStyle" onchange="changeFont()">
```

```
  }

  <option value="Arial">Arial</option>

  <option value="Courier New">Courier New</option>

  <option value="Georgia">Georgia</option>

  <option value="Tahoma">Tahoma</option>

</select>


<p id="fontPara">Change my font using the dropdown!</p>


<script>

  function changeFont() {

    const font = document.getElementById("fontStyle").value;

    document.getElementById("fontPara").style.fontFamily = font;

  }

</script> */


// 10 [Pulse Effect on Click]

/* <style>

  .pulse {

    animation: pulseAnim 0.6s;

  }


  @keyframes pulseAnim {

    0%   { transform: scale(1); }

    50%  { transform: scale(1.1); }

    100% { transform: scale(1); }

  }
```

```
</style>


<button onclick="pulseEffect(this)">Click Me</button>


<script>
  function pulseEffect(btn) {
    btn.classList.remove("pulse"); // reset if already there
    void btn.offsetWidth; // trigger reflow
    btn.classList.add("pulse");
  }
</script> */


// --------------------------------------
// Chapter # 56 Target all elements by tag name
// --------------------------------------


// 1 [Method]
// document.getElementsByTagName("tagname")


// 2 [Target All <p> Elements and Change Color]
/* <p>This is paragraph 1.</p>
<p>This is paragraph 2.</p>
<p>This is paragraph 3.</p>


<button onclick="colorAllP()">Make All Red</button>
```

```
<script>
  function colorAllP() {
    const allParas = document.getElementsByTagName("p");
    for (let i = 0; i < allParas.length; i++) {
      allParas[i].style.color = "red";
    }
  }
</script> */

// 3 [Target All <img> and Resize]
/* <img src="https://via.placeholder.com/100" />
<img src="https://via.placeholder.com/100" />
<img src="https://via.placeholder.com/100" />

<br><br>
<button onclick="resizeImages()">Resize All Images</button>

<script>
  function resizeImages() {
    const allImages = document.getElementsByTagName("img");
    for (let i = 0; i < allImages.length; i++) {
      allImages[i].style.width = "150px";
      allImages[i].style.border = "2px solid green";
    }
  }
</script> */
```

```
// 4 [Hide All <li> Items]

/* <ul>

  <li>Apple</li>

  <li>Mango</li>

  <li>Banana</li>

</ul>


<button onclick="hideList()">Hide All List Items</button>


<script>

  function hideList() {

    const items = document.getElementsByTagName("li");

    for (let i = 0; i < items.length; i++) {

      items[i].style.display = "none";

    }

  }

</script> */


// 5 [Add Numbering to All <h2> Tags]

/* <h2>Introduction</h2>

<h2>Features</h2>

<h2>Contact</h2>


<button onclick="numberHeadings()">Add Numbers</button>
```

```
<script>
  function numberHeadings() {
    const headings = document.getElementsByTagName("h2");
    for (let i = 0; i < headings.length; i++) {
      headings[i].innerText = (i + 1) + ". " + headings[i].innerText;
    }
  }
</script> */

// 6 [Add Border to All Images]
/* <img src="https://via.placeholder.com/100" />
<img src="https://via.placeholder.com/100" />
<img src="https://via.placeholder.com/100" />

<button onclick="borderImages()">Border All</button>

<script>
  function borderImages() {
    const imgs = document.getElementsByTagName("img");
    for (let img of imgs) {
      img.style.border = "3px dashed red";
      img.style.margin = "10px";
    }
  }
</script> */
```

```
// --------------------------------------
// Chapter # 57 Target some elements by tag name
// --------------------------------------

// 1 [Change Only First and Last <p> Color]
/* <p>Paragraph 1</p>
<p>Paragraph 2</p>
<p>Paragraph 3</p>

<button onclick="styleSome()">Style First & Last</button>

<script>
  function styleSome() {
    const paras = document.getElementsByTagName("p");
    paras[0].style.color = "blue"; // first
    paras[paras.length - 1].style.color = "green"; // last
  }
</script> */

// 2 [Change Only Even <li> Items]
/* <ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
```

```
<button onclick="styleEven()">Even Only</button>

<script>
  function styleEven() {
    const items = document.getElementsByTagName("li");
    for (let i = 0; i < items.length; i++) {
      if (i % 2 === 1) { // 1 = second, 3 = fourth, etc.
        items[i].style.background = "#ffe0b2";
      }
    }
  }
</script> */

// 3 [Hide Middle Paragraph (if total = 3)]
/* <p>Intro</p>
<p>Main Content</p>
<p>Conclusion</p>

<button onclick="hideMiddle()">Hide Middle</button>

<script>
  function hideMiddle() {
    const paras = document.getElementsByTagName("p");
    if (paras.length >= 3) {
      paras[1].style.display = "none"; // middle paragraph
    }
  }
</script> */

// 4 [Add Star to Last 2 List Items Only]
/* <ul>
  <li>Option A</li>
  <li>Option B</li>
  <li>Option C</li>
  <li>Option D</li>
</ul>

<button onclick="markLastTwo()">Mark Last 2</button>

<script>
  function markLastTwo() {
    const listItems = document.getElementsByTagName("li");
    const len = listItems.length;

    listItems[len - 1].innerText += " ⭐";
    listItems[len - 2].innerText += " ⭐";
  }
</script> */

// 5 [Underline First 3 <h3> Headings]
/* <h3>Welcome</h3>
```

```
<h3>About</h3>

<h3>Services</h3>

<h3>Contact</h3>


<button onclick="underlineTop()">Underline Top 3</button>


<script>

  function underlineTop() {

    const h3s = document.getElementsByTagName("h3");

    for (let i = 0; i < 3 && i < h3s.length; i++) {

      h3s[i].style.textDecoration = "underline";

    }

  }

</script> */


// 6 [Add ✨ to Every 3rd <li> Item]

/* <ul>

  <li>Item A</li>

  <li>Item B</li>

  <li>Item C</li>

  <li>Item D</li>

  <li>Item E</li>

  <li>Item F</li>

</ul>


<button onclick="starEveryThird()">Highlight Every 3rd</button>
```

```
<script>
  function starEveryThird() {
    const items = document.getElementsByTagName("li");
    for (let i = 2; i < items.length; i += 3) {
      items[i].innerText += " ✨";
      items[i].style.fontWeight = "bold";
    }
  }
</script> */

// 7 [Replace Certain Word in All Paragraphs]
/* <p>I love cats.</p>
<p>Cats are amazing animals.</p>
<p>Dogs are cool too.</p>

<button onclick="replaceWord()">Replace 'Cats' with 'Dogs'</button>

<script>
  function replaceWord() {
    const paras = document.getElementsByTagName("p");
    for (let p of paras) {
      p.innerText = p.innerText.replace(/cats/gi, "dogs");
    }
  }
</script> */
```

```
// 8 [Detect Empty Tags and Highlight Them]

/* <p></p>

<p>Hello World</p>

<p> </p>


<button onclick="highlightEmpty()">Highlight Empty</button>


<script>

  function highlightEmpty() {

    const paras = document.getElementsByTagName("p");

    for (let p of paras) {

      if (p.innerText.trim() === "") {

        p.style.backgroundColor = "yellow";

      }

    }

  }

</script> */


// 9 [Add IDs Dynamically to All Paragraphs]

/* <p>Intro</p>

<p>Body</p>

<p>Summary</p>


<button onclick="addIDs()">Add IDs</button>
```

```
}

<script>
  function addIDs() {
    const paras = document.getElementsByTagName("p");
    for (let i = 0; i < paras.length; i++) {
      paras[i].id = "para" + (i + 1);
    }
    alert("IDs added like para1, para2, para3...");
  }
</script> */

// 10 [Add IDs Dynamically and Show Them in Text]
/* <p>Paragraph without ID</p>
<p>Another paragraph</p>
<p>Yet another one</p>

<button onclick="addIDs()">Add IDs and Show</button>

<script>
  function addIDs() {
    const paras = document.getElementsByTagName("p");
    for (let i = 0; i < paras.length; i++) {
      const idName = "para" + (i + 1);
      paras[i].id = idName;
      paras[i].innerText = `ID: ${idName} — ` + paras[i].innerText;
    }
  }
```

```
</script> */

// --------------------------------------
// Chapter # 58 [The DOM]
// --------------------------------------

// Definition
// DOM ek tree-like structure hai jo HTML document ko represent karta hai.
// har ek HTML tag ek node hota hai (document → html → head/body → elements → attributes → text).
// JavaScript ke through hum DOM ke nodes ko access aur manipulate kar sakte hain.

// Syntax (Access karna)
// document.getElementById("idName")
// document.getElementsByClassName("className")
// document.getElementsByTagName("tagName")
// document.querySelector("cssSelector")
// document.querySelectorAll("cssSelector")

// 1 [Get element by ID]
// let title = document.getElementById("mainTitle");
// title.innerText = "Hello DOM!";

// 2 [Get element by Class]
// let items = document.getElementsByClassName("list-item");
// items[0].style.color = "red";
// 3 [Get element by Tag]
// let paras = document.getElementsByTagName("p");
// console.log(paras.length);

// 4 [querySelector]
// let btn = document.querySelector("#submitBtn");
// btn.style.background = "blue";

// 5 [querySelectorAll]
// let allBtns = document.querySelectorAll(".btn");
// allBtns.forEach(b => b.style.margin = "10px");

// 6 [Changing innerHTML]
// document.getElementById("content").innerHTML = "<b>New Content</b>";

// 7 [Changing Attributes]
// document.getElementById("img").setAttribute("src", "newImage.jpg");

// 8 [Creating Elements]
// let newDiv = document.createElement("div");
// newDiv.innerText = "Hello New Div!";
// document.body.appendChild(newDiv);

// 9 [Removing Elements]
// let para = document.getElementById("removeMe");
// para.remove();

// 10 [Adding Event Listener via DOM]
// document.getElementById("btn").addEventListener("click", () => {
//   alert("Button clicked via DOM!");
// });

// Real Life Uses :
// 1.Button click hone par text change karna
// 2.Form fill karne ke baad validation
// 3.Dynamic content add/remove (e-commerce products, todo list)
// 4.Images aur attributes update karna
// 5.Responsive UI build karna

// --------------------------------------
// Chapter # 59 [The DOM: Parents and children]
// --------------------------------------

// Parents & Children Concept :
// Parent Node: jiske andar element hota hai.
// Child Node: jo element parent ke andar hota hai.
// HTML me <div> ke andar <p> ho to <div> parent aur <p> child hai.

// Important Properties / Methods :
// 1.parentNode → kisi element ka parent dikhata hai.
// 2.children → sirf element children (no text nodes).
// 3.childNodes → sari child nodes (text, comment, element sab).

// 4.firstElementChild → pehla child element.

// 5.lastElementChild → akhri child element.

// 6.nextElementSibling → agla bhai element (same parent).

// 7.previousElementSibling → pichla bhai element.


// 1 [Access parentNode]

/* <div id="parent">

  <p id="child">Hello</p>

</div>

<script>

let c = document.getElementById("child");

console.log(c.parentNode.id); // parent

</script> */


// 2 [Access children]

/* <ul id="list">

  <li>Apple</li>

  <li>Mango</li>

  <li>Banana</li>

</ul>

<script>

let ul = document.getElementById("list");

console.log(ul.children[0].innerText); // Apple

</script> */
```

```
// 3 [Using childNodes (text + comment + elements)]

/* <div id="box">

  <p>One</p>

  <p>Two</p>

</div>

<script>

let box = document.getElementById("box");

console.log(box.childNodes.length); // includes text nodes

</script> */


// 4 [First and Last child]

// let ul = document.getElementById("list");

// console.log(ul.firstElementChild.innerText); // Apple

// console.log(ul.lastElementChild.innerText);  // Banana


// 5 [Next & Previous sibling]

/* <ul>

  <li id="mango">Mango</li>

  <li>Banana</li>

</ul>

<script>

let mango = document.getElementById("mango");

console.log(mango.nextElementSibling.innerText); // Banana

</script> */


// 6 [Traverse from child → parent → child]
```

```
/* <div id="container">
  <ul>
    <li id="child1">One</li>
    <li>Two</li>
  </ul>
</div>
<script>
let child = document.getElementById("child1");
console.log(child.parentNode.parentNode.id); // container
</script> */

// --------------------------------------
// Chapter # 60 [The DOM: Finding children]
// --------------------------------------

// Main Properties & Methods :
// children → sirf element nodes return karta hai (text/comment ignore).
// childNodes → sari nodes (text, comment, element sab) return karta hai.
// firstElementChild → parent ka pehla child element.
// lastElementChild → parent ka last child element.
// querySelector/querySelectorAll → parent ke andar specific child find karna.

// 1 [Get all children using .children]
/* <ul id="fruits">
  <li>Apple</li>
  <li>Mango</li>
  <li>Banana</li>

</ul>

<script>

let ul = document.getElementById("fruits");

console.log(ul.children);       // HTMLCollection

console.log(ul.children[1].innerText); // Mango

</script> */


// 2 [Using .childNodes (text + comments bhi)]

/* <div id="box">

  <p>One</p>

  <p>Two</p>

</div>

<script>

let box = document.getElementById("box");

console.log(box.childNodes.length); // text nodes bhi count honge

</script> */


// 3 [First and last child]

// let ul = document.getElementById("fruits");

// console.log(ul.firstElementChild.innerText); // Apple

// console.log(ul.lastElementChild.innerText);  // Banana


// 4 [Loop through children]

// for (let child of ul.children) {

//   console.log("Fruit: " + child.innerText);
// }


// 5 [Find specific child inside parent]

/* <div id="container">

  <p class="info">First</p>

  <p class="info">Second</p>

</div>

<script>

let container = document.getElementById("container");

let second = container.querySelectorAll(".info")[1];

console.log(second.innerText); // Second

</script> */


// ---------------------------------------

// Chapter # 61 [The DOM: Junk artifacts and nodeType]

// ---------------------------------------


// Junk Artifacts kya hote hain? :

// jab hum .childNodes use karte hain to sirf elements hi nahi, balki:

// whitespace (line breaks, tabs)

// text nodes (jo blank space hote hain)

// comment nodes

// sab aajate hain → yehi “junk artifacts” kehlate hain.


// nodeType Property

// nodeType ek number return karta hai jo node ka type batata hai.
// Common Values:
// 1 → Element Node (<div>, <p>, <li>, etc.)
// 3 → Text Node (spaces, text content)
// 8 → Comment Node (<!-- comment -->)

// 1 [Junk artifacts with childNodes]
/* <ul id="list">
  <li>Apple</li>
  <li>Mango</li>
  <li>Banana</li>
</ul>
<script>
let ul = document.getElementById("list");
console.log(ul.childNodes);
// Text nodes (whitespace) bhi count honge
</script> */

// 2 [Filtering by nodeType]
// ul.childNodes.forEach(node => {
//   if (node.nodeType === 1) { // element node
//     console.log("Element: " + node.innerText);
//   }
// });

// 3 [Detect text node]
// ul.childNodes.forEach(node => {
//   if (node.nodeType === 3) {
//     console.log("Text node found (junk)");
//   }
// });

// 4 [Detect comment node]
/* <div id="box">
  <!-- This is a comment -->
  <p>Hello</p>
</div>
<script>
let box = document.getElementById("box");
box.childNodes.forEach(node => {
  if (node.nodeType === 8) {
    console.log("Comment node found");
  }
});
</script> */

// 5 [Cleaner alternative: .children]
// console.log(ul.children);
// // sirf elements aayenge, junk artifacts nahi

// --------------------------------------
// Chapter # 62 [ The DOM: More ways to target elements]
// --------------------------------------

// Common Selection Methods :
// 1.getElementById → ek element by ID
// 2.getElementsByClassName → elements by class
// 3.getElementsByTagName → elements by tag name
// 4.querySelector → CSS selector se ek element
// 5.querySelectorAll → CSS selector se multiple elements

// 1 [getElementById]
/* <p id="msg">Hello World</p>
<script>
let el = document.getElementById("msg");
console.log(el.innerText);  // Hello World
</script> */

// 2 [getElementsByClassName]
/* <p class="note">First Note</p>
<p class="note">Second Note</p>
<script>
let notes = document.getElementsByClassName("note");
console.log(notes[0].innerText); // First Note
console.log(notes[1].innerText); // Second Note
</script> */

// 3 [getElementsByTagName]
```

```
/* <ul>
  <li>Apple</li>
  <li>Mango</li>
</ul>
<script>
let items = document.getElementsByTagName("li");
for (let item of items) {
  console.log(item.innerText);
}
</script> */

// 4 [querySelector (first match only)]
/* <div>
  <p class="info">One</p>
  <p class="info">Two</p>
</div>
<script>
let first = document.querySelector(".info");
console.log(first.innerText); // One
</script> */

// 5 [querySelectorAll (all matches)]
/* <div>
  <p class="info">One</p>
  <p class="info">Two</p>
</div>
```

```
<script>

let all = document.querySelectorAll(".info");

all.forEach(el => console.log(el.innerText));

</script> */


// 6 [Select nested elements]

// let container = document.querySelector("div");

// let para = container.querySelector("p");

// console.log(para.innerText);


// 7 [Select by attribute]

/* <input type="text" name="username">

<script>

let userInput = document.querySelector("input[name='username']");

console.log(userInput);

</script> */


// 8 [Select nth-child]

/* <ul>

  <li>One</li>

  <li>Two</li>

  <li>Three</li>

</ul>

<script>

let second = document.querySelector("ul li:nth-child(2)");

console.log(second.innerText); // Two
```

```
</script> */

// 9 [Select with multiple selectors]
// let el = document.querySelector("p.info, span.note");
// console.log(el);

// 10 [Using IDs inside querySelector]
// let el = document.querySelector("#msg");
// console.log(el.innerText);

// --------------------------------------
// Chapter # 63 [The DOM: Getting a target's name]
// --------------------------------------

// Important Concept :
// event.target → wo element return karta hai jis par event trigger hua.
// event.target.name → agar us element me name attribute hoga to uski value milegi.
// Useful jab forms, inputs, radio buttons, dropdowns ke sath kaam karna ho.

// 1 [Input field name]
/* <input type="text" name="username" placeholder="Enter username">

<script>
document.querySelector("input").addEventListener("input", function(e) {
  console.log("Target name: " + e.target.name); // username
});
```

```
</script> */

// 2 [Multiple inputs]
/* <input type="text" name="firstName" placeholder="First Name">
<input type="text" name="lastName" placeholder="Last Name">

<script>
document.querySelectorAll("input").forEach(inp => {
  inp.addEventListener("focus", e => {
    console.log("Focused input name: " + e.target.name);
  });
});
</script> */

// 3 [Radio buttons]
/* <form>
  <input type="radio" name="gender" value="male"> Male
  <input type="radio" name="gender" value="female"> Female
</form>

<script>
document.querySelectorAll("input[type=radio]").forEach(radio => {
  radio.addEventListener("change", e => {
    console.log("Target name: " + e.target.name); // gender
    console.log("Selected value: " + e.target.value);
  });
});
</script> */

// 4 [Dropdown select]
/* <select name="country">
  <option>Pakistan</option>
  <option>India</option>
</select>

<script>
document.querySelector("select").addEventListener("change", e => {
  console.log("Target name: " + e.target.name); // country
});
</script> */

// 5 [Button click]
/* <button name="loginBtn">Login</button>
<button name="signupBtn">Signup</button>

<script>
document.querySelectorAll("button").forEach(btn => {
  btn.addEventListener("click", e => {
    console.log("Clicked button name: " + e.target.name);
  });
});
</script> */
```

// --------------------------------------

// Chapter # 64 [The DOM: Counting elements]

// --------------------------------------

// Common Methods :

// .length → jab collection milta hai (getElementsByTagName, getElementsByClassName, querySelectorAll)

// childElementCount → parent ke andar kitne element children hain

// children.length → child elements ka count

// childNodes.length → saare nodes ka count (junk artifacts bhi include)

// 1 [Count all paragraphs]

/* <p>One</p>

<p>Two</p>

<p>Three</p>

<script>

let paras = document.getElementsByTagName("p");

console.log("Total paragraphs: " + paras.length); // 3

</script> */

// 2 [Count elements by class]

/* <div class="note">Note 1</div>

<div class="note">Note 2</div>

<script>

```
let notes = document.getElementsByClassName("note");

console.log("Total notes: " + notes.length); // 2

</script> */


// 3 [Count with querySelectorAll]

/* <ul>

  <li>Apple</li>

  <li>Mango</li>

  <li>Banana</li>

</ul>

<script>

let items = document.querySelectorAll("ul li");

console.log("Total items: " + items.length); // 3

</script> */


// 4 [Count children of a parent]

/* <div id="box">

  <p>One</p>

  <p>Two</p>

</div>

<script>

let box = document.getElementById("box");

console.log("Child elements: " + box.childElementCount); // 2

</script> */


// 5 [Count including junk artifacts (childNodes)]
```

```
/* <div id="container">
  <p>One</p>
  <p>Two</p>
</div>
<script>
let cont = document.getElementById("container");
console.log("Child nodes: " + cont.childNodes.length); // whitespace bhi count hoga
</script> */

// --------------------------------------
// Chapter # 65 []
// --------------------------------------

// Important Methods :
// getAttribute("attr") → attribute ki value read karne ke liye
// setAttribute("attr","value") → attribute set/update karne ke liye
// removeAttribute("attr") → attribute delete karne ke liye
// hasAttribute("attr") → check karne ke liye attribute exist karta hai ya nahi
// element.id / element.className / element.src → direct property access

// 1 [Get attribute]
/* <a id="myLink" href="https://google.com">Google</a>
<script>
let link = document.getElementById("myLink");
console.log(link.getAttribute("href")); // https://google.com
</script> */
```

```
// 2 [Set/Update attribute]

// link.setAttribute("href", "https://youtube.com");

// console.log(link.getAttribute("href")); // https://youtube.com


// 3 [Remove attribute]

// link.removeAttribute("href");

// console.log(link.getAttribute("href")); // null


// 4 [Check attribute existence]

// console.log(link.hasAttribute("href")); // false (remove karne ke baad)


// 5 [Direct property access]

/* <img id="logo" src="logo.png" alt="Website Logo">

<script>

let img = document.getElementById("logo");

console.log(img.src);        // full URL path

img.src = "new-logo.png";    // change image

console.log(img.alt);        // Website Logo

</script> */


// 6 [Add custom attribute]

// link.setAttribute("data-user", "Hasnain");

// console.log(link.getAttribute("data-user")); // Hasnain


// 7 [Class attribute special handling]
// link.setAttribute("class", "btn btn-primary");

// console.log(link.className); // btn btn-primary


// 8 [Toggle attribute dynamically]

/* <input type="text" id="inp" placeholder="Enter name">

<script>

let inp = document.getElementById("inp");

inp.setAttribute("disabled", true); // disable field

setTimeout(() => inp.removeAttribute("disabled"), 3000); // enable after 3s

</script> */


// ---------------------------------------

// Chapter # 66 [The DOM: Attribute names and values]

// ---------------------------------------


// Important Properties & Methods :

// getAttribute(attrName) → attribute ki value nikalne ke liye

// setAttribute(attrName, value) → attribute ki value set/replace karne ke liye

// removeAttribute(attrName) → attribute ko delete karne ke liye

// hasAttribute(attrName) → check karne ke liye attribute exist karta hai ya nahi

// attributes → ek NamedNodeMap return karta hai jisme saare attribute names aur values hoti
hain


// 1 [Get attribute name and value]

/* <a id="myLink" href="https://google.com" target="_blank">Google</a>

<script>
```

```
let link = document.getElementById("myLink");

console.log(link.getAttribute("href"));   // https://google.com

console.log(link.getAttribute("target")); // _blank

</script> */


// 2 [Set attribute value]

// link.setAttribute("href", "https://youtube.com");

// console.log(link.getAttribute("href")); // https://youtube.com


// 3 [Remove attribute]

// link.removeAttribute("target");

// console.log(link.hasAttribute("target")); // false


// 4 [Access all attributes]

/* <img id="logo" src="logo.png" alt="Website Logo" width="200">

<script>

let img = document.getElementById("logo");

for (let attr of img.attributes) {

  console.log(attr.name + " = " + attr.value);

}

// Output: id=logo, src=logo.png, alt=Website Logo, width=200

</script> */


// 5 [Check if attribute exists]

// console.log(img.hasAttribute("alt"));   // true

// console.log(img.hasAttribute("title")); // false
// 6 [Custom data attributes]
/* <button id="btn" data-user="Hasnain" data-role="admin">Click</button>
<script>
let btn = document.getElementById("btn");
console.log(btn.getAttribute("data-user")); // Hasnain
console.log(btn.getAttribute("data-role")); // admin
</script> */

// 7 [Change attribute dynamically]
// btn.setAttribute("data-role", "editor");
// console.log(btn.getAttribute("data-role")); // editor

// --------------------------------------
// Chapter # 67 [The DOM: Adding nodes]
// --------------------------------------

// Important Methods :
// document.createElement("tag") → naya element banata hai
// document.createTextNode("text") → naya text node banata hai
// parent.appendChild(node) → parent ke andar node add karta hai (last me)
// parent.insertBefore(newNode, existingNode) → specific jagah insert karna
// element.append() / element.prepend() → naya element start ya end me add karna (modern
way)

// 1 [Create and append element]
```

```
/* <ul id="fruits">

  <li>Apple</li>

</ul>

<script>

let ul = document.getElementById("fruits");

let newItem = document.createElement("li");

newItem.innerText = "Mango";

ul.appendChild(newItem); // Mango add ho gaya

</script> */


// 2 [Add text node]

// let textNode = document.createTextNode("Banana");

// ul.appendChild(textNode); // text as a node add ho gaya


// 3 [Insert before specific node]

// let firstItem = ul.firstElementChild;

// let newNode = document.createElement("li");

// newNode.innerText = "Orange";

// ul.insertBefore(newNode, firstItem); // Orange Apple se pehle


// 4 [Using append() and prepend()]

// let grape = document.createElement("li");

// grape.innerText = "Grapes";

// ul.append(grape);   // end me add


// let cherry = document.createElement("li");
// cherry.innerText = "Cherry";

// ul.prepend(cherry); // start me add


// 5 [Add multiple nodes at once]

// let div = document.createElement("div");

// div.innerHTML = "<p>First para</p><p>Second para</p>";

// document.body.appendChild(div);


// --------------------------------------

// Chapter # 68 [The DOM: Inserting nodes]

// --------------------------------------


// Important Methods :

// appendChild(node) → parent ke end me add karta hai

// insertBefore(newNode, referenceNode) → reference node se pehle insert karta hai

// append(node1, node2, …) → multiple nodes ko end me add kar sakte ho

// prepend(node) → parent ke start me add karta hai

// insertAdjacentElement(position, element) → precise jagah insert karta hai


// 1 [AppendChild (end me insert karna)]

/* <ul id="fruits">

  <li>Apple</li>

</ul>

<script>

let ul = document.getElementById("fruits");

let li = document.createElement("li");
```

```
li.innerText = "Mango";

ul.appendChild(li); // Mango last me add

</script> */


// 2 [Insert before specific node]

// let orange = document.createElement("li");

// orange.innerText = "Orange";

// let firstItem = ul.firstElementChild;

// ul.insertBefore(orange, firstItem); // Orange Apple se pehle


// 3 [Prepend (start me insert karna)]

// let banana = document.createElement("li");

// banana.innerText = "Banana";

// ul.prepend(banana); // Banana sabse pehle


// 4 [Append multiple nodes]

// let li1 = document.createElement("li");

// li1.innerText = "Cherry";


// let li2 = document.createElement("li");

// li2.innerText = "Grapes";


// ul.append(li1, li2); // dono end me add ho gaye


// 5 [insertAdjacentElement (exact position)]

/* <div id="box">
```

```
  <p>First</p>
</div>
<script>
let box = document.getElementById("box");
let para = document.createElement("p");
para.innerText = "Inserted para";

// 4 positions: beforebegin, afterbegin, beforeend, afterend
box.insertAdjacentElement("beforeend", para); // <div> ke andar last me
</script> */

// --------------------------------------
// Chapter # 69 [Objects]
// --------------------------------------

// 1 [syntax]
// let person = {
//   name: "Hasnain",
//   age: 22,
//   city: "Karachi"
// };

// --------------------------------------
// Chapter # 70 [Objects: Properties]
// --------------------------------------
// Object
// let person = {
//   name: "Hasnain",
//   age: 22,
//   city: "Karachi"
// };

// 1 [Accessing Object Properties]
// console.log(person.name);    // dot notation
// console.log(person["age"]);  // bracket notation

// 2 [Adding & Updating Values]
// person.country = "Pakistan";   // new property add
// person.age = 23;              // update property

// 3 [Deleting Property]
// delete person.city;
// console.log(person);

// 4 [Check if Property Exists]
// console.log("name" in person);   // true
// console.log("salary" in person); // false

// --------------------------------------
// Chapter # 71 [Objects: Methods]
// --------------------------------------
// 1 [Basic Method]

// let person = {

//   name: "Hasnain",

//   age: 22,


//   // Method

//   greet: function () {

//     return `Hello, my name is ${this.name} and I am ${this.age} years old.`;

//   }

// }


// 2 [Short Method Syntax (ES6+)]

// let person = {

//   name: "Ali",

//   sayHi() {

//     console.log(`Hi, ${this.name}!`);

//   }

// };


// person.sayHi();  // Hi, Ali!


// 3 [Object.keys(obj)]

// let car = { brand: "Toyota", model: "Corolla", year: 2020 };

// console.log(Object.keys(car));  // ["brand", "model", "year"]
// 4 [Object.values(obj)]

// let car = { brand: "Toyota", model: "Corolla", year: 2020 };

// console.log(Object.values(car)); //["Toyota", "Corolla", 2020]


// 5 [Object.entries(obj)]

// let car = { brand: "Toyota", model: "Corolla", year: 2020 };

// console.log(Object.entries(car)); // [["brand", "Toyota"], ["model", "Corolla"], ["year", 2020]]


// 6 [Object.assign(target, source)]

// let obj1 = { a: 1 };

// let obj2 = { b: 2 };

// let merged = Object.assign({}, obj1, obj2);


// console.log(merged); // { a: 1, b: 2 }


// 7 [Object.freeze(obj)]

// let user = { id: 101, role: "admin" };

// Object.freeze(user);

// user.role = "user";  // change nahi hoga

// console.log(user);   // { id: 101, role: "admin" }


// 8 [Object.seal(obj)]

// let stu = { name: "Sara", age: 20 };

// Object.seal(stu);

// stu.age = 21;       // update hoga

// delete stu.name;     // delete nahi hoga
// console.log(stu);    // { name: "Sara", age: 21 }


// --------------------------------------

// Chapter # 72 [Objects: Constructors]

// --------------------------------------


// 1 [Basic Constructor Function Syntax]

// function Person(name, age) {

//   this.name = name;   // property

//   this.age = age;     // property


//   this.greet = function() {   // method

//     console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);

//   };

// }


// // New objects banane ke liye "new" keyword use hota hai

// let person1 = new Person("Hasnain", 22);

// let person2 = new Person("Ali", 25);


// person1.greet();  // Hello, my name is Hasnain and I am 22 years old.

// person2.greet();  // Hello, my name is Ali and I am 25 years old.


// 2 [Constructor with Prototype]

// function Car(brand, model) {

//   this.brand = brand;
//   this.model = model;
// }

// // method ko prototype par add karna best practice hai
// Car.prototype.start = function() {
//   console.log(`${this.brand} ${this.model} is starting...`);
// };

// let car1 = new Car("Toyota", "Corolla");
// let car2 = new Car("Honda", "Civic");

// car1.start();  // Toyota Corolla is starting...
// car2.start();  // Honda Civic is starting...

// 3 [Constructor using ES6 Class (Modern Way)]
// class Student {
//   constructor(name, roll) {
//     this.name = name;
//     this.roll = roll;
//   }

//   // method
//   display() {
//     console.log(`Student: ${this.name}, Roll No: ${this.roll}`);
//   }
// }
// let s1 = new Student("Sara", 101);
// let s2 = new Student("Ahmed", 102);

// s1.display();  // Student: Sara, Roll No: 101
// s2.display();  // Student: Ahmed, Roll No: 102

// --------------------------------------
// Chapter # 73 [Objects: Constructors for methods]
// --------------------------------------

// 1 [Constructor with Methods (Direct Inside Function)]
// function Person(name, age) {
//   this.name = name;
//   this.age = age;

//   // method constructor ke andar define kiya
//   this.sayHello = function () {
//     console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
//   };
// }

// let p1 = new Person("Hasnain", 22);
// let p2 = new Person("Ali", 25);

// p1.sayHello();  // Hello, my name is Hasnain and I am 22 years old.
// p2.sayHello();  // Hello, my name is Ali and I am 25 years old.


// 2 [Constructor + Prototype Methods (Efficient Way)]

// function Car(brand, model) {

//   this.brand = brand;

//   this.model = model;

// }


// // method ko prototype me add karo

// Car.prototype.drive = function () {

//   console.log(`${this.brand} ${this.model} is driving...`);

// };


// let c1 = new Car("Toyota", "Corolla");

// let c2 = new Car("Honda", "Civic");


// c1.drive();  // Toyota Corolla is driving...

// c2.drive();  // Honda Civic is driving...


// 3 [Constructor Methods using ES6 Class]

// class Student {

//   constructor(name, roll) {

//     this.name = name;

//     this.roll = roll;

//   }
//   // yeh method class ke andar likha gaya
//   display() {
//     console.log(`Student: ${this.name}, Roll No: ${this.roll}`);
//   }
// }

// let s1 = new Student("Sara", 101);
// let s2 = new Student("Ahmed", 102);

// s1.display();  // Student: Sara, Roll No: 101
// s2.display();  // Student: Ahmed, Roll No: 102

// 4 []

// --------------------------------------
// Chapter # 74 [Objects: Prototypes]
// --------------------------------------

// 1 [Prototype Basic Example]
// let person = {
//   greet: function () {
//     console.log("Hello World!");
//   }
// };

// let user = Object.create(person);
// user.name = "Hasnain";

// console.log(user.name);  // Hasnain
// user.greet();           // Hello World!

// 2 [Function Constructors + Prototype]
// function Animal(type) {
//   this.type = type;
// }

// // method ko prototype me add kiya
// Animal.prototype.speak = function () {
//   console.log(`${this.type} makes a sound`);
// };

// let dog = new Animal("Dog");
// let cat = new Animal("Cat");

// dog.speak();  // Dog makes a sound
// cat.speak();  // Cat makes a sound

// 3 [Checking Prototype Chain]
// function Car(brand) {
//   this.brand = brand;
// }
// Car.prototype.drive = function () {
//   console.log(`${this.brand} is driving`);
// };

// let c1 = new Car("Toyota");

// console.log(Object.getPrototypeOf(c1) === Car.prototype);  // true

// 4 [Inheriting Methods]
// function Animal(name) {
//   this.name = name;
// }
// Animal.prototype.speak = function() {
//   console.log(this.name + " makes a sound.");
// };

// function Dog(name) {
//   Animal.call(this, name);
// }
// Dog.prototype = Object.create(Animal.prototype);

// let dog = new Dog("Tommy");
// dog.speak();

// --------------------------------------
// Chapter # 75 [Objects: Checking for properties and methods]
// --------------------------------------

// 1 [Checking Properties (in operator)]
// let car = {
//   brand: "Toyota",
//   model: "Corolla",
//   year: 2020
// };

// console.log("brand" in car);  // true
// console.log("color" in car);  // false

// 2 [Using hasOwnProperty()]
// let user = {
//   name: "Hasnain",
//   age: 21
// };

// console.log(user.hasOwnProperty("name"));   // true
// console.log(user.hasOwnProperty("email"));  // false

// 3 [Checking Methods]
// let student = {
//   name: "Ali",
//   study: function() {
//     return "Studying...";
//   }
// };

// console.log("study" in student);                // true
// console.log(typeof student.study === "function"); // true

// 4 [Check property with undefined]
// let book = {
//   title: "JavaScript Basics",
//   pages: 200
// };

// if (book.author === undefined) {
//   console.log("Author not found");
// } else {
//   console.log("Author:", book.author);
// }

// 5 [Check multiple properties]
// let phone = {
//   brand: "Samsung",
//   price: 50000,
//   call: function() {
//     return "Calling...";
//   }
// };
// ["brand", "camera", "price"].forEach(prop => {
//   if (prop in phone) {
//     console.log(prop + " exists ✅");
//   } else {
//     console.log(prop + " not found ❌");
//   }
// });

// 6 [Method existence check]
// let person = {
//   name: "Hasnain",
//   greet: function() {
//     return "Hello!";
//   }
// };

// if (typeof person.greet === "function") {
//   console.log("Method 'greet' is available ✅");
// } else {
//   console.log("Method 'greet' not found ❌");
// }

// 7 [Safe property access with ?.]
// let laptop = {
//   brand: "HP",
//   details: {
//     ram: "16GB"
//   }
// };

// console.log(laptop.details?.ram);     // "16GB"
// console.log(laptop.details?.storage); // undefined (safe check)


// ---------------------------------------
// Chapter # 76 [ Browser control: Getting and setting the URL]
// ---------------------------------------

// 1 [window.location.href (Get URL)]
// console.log(window.location.href);

// 2 [window.location.href (Set URL / Redirect)]
// window.location.href = "https://google.com";

// 3 [window.location.assign()]
// window.location.assign("https://youtube.com");

// 4 [window.location.replace()]
// window.location.replace("https://github.com");

// 5 [window.location.reload()]
// window.location.reload();
// 6 [window.open()]
// window.open("https://google.com", "_blank");

// 7 [setTimeout()]
// setTimeout(() => {
//   console.log("Hello after 3 seconds");
// }, 3000);

// 8 [setInterval()]
// setInterval(() => {
  // console.log("Repeating every 2 seconds");
// }, 2000);

// 9 [history.back() & history.forward()]
// // Go back
// history.back();

// // Go forward
// history.forward();

// ---------------------------------------
// Chapter # 77 [Browser control: Getting and setting the URL another way]
// ---------------------------------------

// 1 [window.location.host]
// console.log(window.location.host);

// 2 [window.location.hostname]
// console.log(window.location.hostname);

// 3 [window.location.pathname]
// console.log(window.location.pathname);

// 4 [window.location.search]
// console.log(window.location.search);

// 5 [URLSearchParams (modern way for query string)]
// let params = new URLSearchParams(window.location.search);
// console.log(params.get("id"));

// Summary:
// host → domain + port
// hostname → sirf domain
// pathname → page ka path
// search → query string
// URLSearchParams → query string se values nikalna easy way

// --------------------------------------
// Chapter # 78 [Browser control: Forward and reverse]
// --------------------------------------
// 1 [history.back()]

// history.back();


// 2 [history.forward()]

// history.forward();


// 3 [history.go(-1)]

// history.go(-1);


// 4 [history.go(1)]

// history.go(1);


// 5 [history.go(n)]

// history.go(-2); // 2 steps back

// history.go(3);  // 3 steps forward


// ---------------------------------------

// Chapter # 79 [Browser control: Filling the window with content]

// ---------------------------------------


// 1 [document.write()]

// document.writeln("<h1>Hello, World!</h1>");


// 2 [document.body.innerHTML]

// document.body.innerHTML = "<h2>Full Page Content Loaded!</h2>";
// 3 [window.open() with document.write()]

// let win = window.open("", "", "width=400,height=300");

// win.document.writeln("<p>This is a new window with custom content</p>");


// 4 [iframe contentWindow]

// let frame = document.getElementById("myFrame");

// frame.contentWindow.document.write("<h3>Iframe Filled with Content</h3>");


// 5 [Using innerText for plain content]

// document.body.innerText = "This will replace whole window with plain text.";


// --------------------------------------

// Chapter # 80 [Browser control: Controlling the window's size and location]

// --------------------------------------


// 1 [window.resizeTo(width, height)]

// let win = window.open("", "", "width=200,height=200");

// win.resizeTo(600, 400);


// 2 [window.resizeBy(x, y)]

// let win = window.open("", "", "width=300,height=200");

// win.resizeBy(200, 100);


// 3 [window.moveTo(x, y)]

// let win = window.open("", "", "width=300,height=200");

// win.moveTo(100, 100);
// 4 [window.moveBy(x, y)]
// let win = window.open("", "", "width=300,height=200");
// win.moveBy(50, 50);

// 5 [window.screen properties]
// console.log(window.screen.width);
// console.log(window.screen.height);

// --------------------------------------
// Chapter # 81 [Browser control: Testing for popup blockers]
// --------------------------------------

// 1 [Basic popup test with window.open()]
// let popup = window.open("", "", "width=200,height=100");
// if (!popup) {
//   console.log("Popup blocked!");
// } else {
//   console.log("Popup allowed!");
//   popup.close();
// }

// 2 [Checking popup.document]
// let popup = window.open("about:blank", "", "width=200,height=100");
// try {
//   if (popup && popup.document) {
```

```
//     console.log("Popup working fine!");
//   }
// } catch (e) {
//   console.log("Popup blocked!");
// }

// 3 [Focus test]
// let popup = window.open("", "", "width=200,height=100");
// if (popup) {
//   popup.focus();
//   console.log("Popup active");
// } else {
//   console.log("Popup blocked");
// }

// 4 [Popup with delay]
// setTimeout(() => {
//   let popup = window.open("", "", "width=200,height=100");
//   if (!popup) {
//     alert("Please allow popups for this site!");
//   }
// }, 1000);

// 5 [Detecting popup close (extra check)]
// let popup = window.open("", "", "width=200,height=100");
// if (popup) {
```

```
//   let timer = setInterval(() => {
//     if (popup.closed) {
//       clearInterval(timer);
//       console.log("Popup closed by user.");
//     }
//   }, 500);
// } else {
//   console.log("Popup blocked!");
// }

// -------------------------------------
// Chapter # 82 [Form validation: text fields]
// -------------------------------------

// 1 [Get and Set dynamically]
/* <input type="text" id="city" placeholder="Enter city">
<button onclick="copyCity()">Copy</button>
<input type="text" id="copyCity">
<script>
function copyCity() {
  let val = document.getElementById("city").value;
  document.getElementById("copyCity").value = val;
}
</script> */

// 2 [Get/Set using form elements]
```

```
/* <form id="myForm">
  <input type="text" name="username" value="DefaultUser">
  <button type="button" onclick="changeUser()">Change</button>
</form>

<script>
function changeUser() {
  let form = document.getElementById("myForm");
  alert("Old Value: " + form.username.value);
  form.username.value = "NewUser123";
}
</script> */

// 3 [Required field check]
/* <input type="text" id="username" placeholder="Enter username">
<button onclick="checkUser()">Submit</button>

<script>
function checkUser() {
  let user = document.getElementById("username").value;
  if (user === "") {
    alert("Username is required!");
  } else {
    alert("Welcome " + user);
  }
}
```

```
</script> */

// 4 [Minimum length check]
/* <input type="text" id="password" placeholder="Enter password">
<button onclick="checkPass()">Submit</button>

<script>
function checkPass() {
  let pass = document.getElementById("password").value;
  if (pass.length < 6) {
    alert("Password must be at least 6 characters!");
  } else {
    alert("Password looks good.");
  }
}
</script> */

// 5 [Email format validation]
/* <input type="text" id="email" placeholder="Enter email">
<button onclick="checkEmail()">Submit</button>

<script>
function checkEmail() {
  let email = document.getElementById("email").value;
  let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
  if (!email.match(pattern)) {
    alert("Invalid email address!");
  } else {
    alert("Email is valid.");
  }
}
</script> */

// 6 [Only letters allowed]
/* <input type="text" id="name" placeholder="Enter your name">
<button onclick="checkName()">Submit</button>

<script>
function checkName() {
  let name = document.getElementById("name").value;
  let letters = /^[A-Za-z ]+$/;
  if (!name.match(letters)) {
    alert("Only letters are allowed!");
  } else {
    alert("Name looks good.");
  }
}
</script> */

// 7 [Using HTML5 attributes]
/* <form>
  <input type="text" name="fullname" placeholder="Enter full name" required minlength="3">
  <input type="email" name="email" placeholder="Enter email" required>

  <button type="submit">Submit</button>

</form> */


// --------------------------------------

// Chapter # 83 [Form validation: drop-downs]

// --------------------------------------


// 1 [Get selected value]

/* <select id="country">

  <option value="pk">Pakistan</option>

  <option value="in">India</option>

  <option value="uk">United Kingdom</option>

</select>

<button onclick="getCountry()">Get Value</button>


<script>

function getCountry() {

  let val = document.getElementById("country").value;

  alert("Selected: " + val);

}

</script> */


// 2 [Set selected value]

/* <select id="lang">

  <option value="en">English</option>
  <option value="ur">Urdu</option>
  <option value="ar">Arabic</option>
</select>
<button onclick="setLang()">Set Urdu</button>

<script>
function setLang() {
  document.getElementById("lang").value = "ur";
}
</script> */

// 3 [Get selected text (not value)]
/* <select id="city">
  <option value="karachi">Karachi</option>
  <option value="lahore">Lahore</option>
  <option value="islamabad">Islamabad</option>
</select>
<button onclick="getCityText()">Get Text</button>

<script>
function getCityText() {
  let sel = document.getElementById("city");
  let text = sel.options[sel.selectedIndex].text;
  alert("Selected Text: " + text);
}
</script> */
```

```
// 4 [Dynamically add new option]
/* <select id="fruits">
  <option value="apple">Apple</option>
  <option value="banana">Banana</option>
</select>
<button onclick="addFruit()">Add Mango</button>

<script>
function addFruit() {
  let sel = document.getElementById("fruits");
  let opt = new Option("Mango", "mango");
  sel.add(opt);
}
</script> */

// 5 [Loop through all options]
/* <select id="cars">
  <option value="honda">Honda</option>
  <option value="toyota">Toyota</option>
  <option value="suzuki">Suzuki</option>
</select>
<button onclick="listCars()">Show All</button>

<script>
function listCars() {
  let sel = document.getElementById("cars");
  let all = [];
  for (let i = 0; i < sel.options.length; i++) {
    all.push(sel.options[i].text);
  }
  alert("Cars: " + all.join(", "));
}
</script> */

// ---------------------------------------
// Chapter # 84 [Form validation: radio buttons]
// ---------------------------------------

// 1 [Get selected value]
/* <form>
  <label><input type="radio" name="gender" value="Male"> Male</label>
  <label><input type="radio" name="gender" value="Female"> Female</label>
  <button type="button" onclick="getGender()">Check</button>
</form>

<script>
function getGender() {
  let val = document.querySelector('input[name="gender"]:checked');
  if (val) {
    alert("Selected: " + val.value);
  } else {
    alert("No option selected!");
  }
}
</script> */

// 2 [Set default checked]
/* <form>
  <label><input type="radio" name="role" value="Student"> Student</label>
  <label><input type="radio" name="role" value="Teacher"> Teacher</label>
</form>

<script>
document.querySelector('input[value="Teacher"]').checked = true;
</script> */

// 3 [Validation (must select one)]
/* <form id="regForm">
  <label><input type="radio" name="plan" value="Free"> Free</label>
  <label><input type="radio" name="plan" value="Premium"> Premium</label>
  <button type="button" onclick="validatePlan()">Submit</button>
</form>

<script>
function validatePlan() {
  let choice = document.querySelector('input[name="plan"]:checked');
  if (!choice) {
    alert("Please select a plan!");
  } else {
    alert("You selected: " + choice.value);
  }
}
</script> */

// 4 [Get all options in group]
/* <form>
  <label><input type="radio" name="color" value="Red"> Red</label>
  <label><input type="radio" name="color" value="Green"> Green</label>
  <label><input type="radio" name="color" value="Blue"> Blue</label>
</form>

<script>
let radios = document.getElementsByName("color");
for (let r of radios) {
  console.log("Option: " + r.value);
}
</script> */

// 5 [Change event listener]
/* <form>
  <label><input type="radio" name="payment" value="Card"> Card</label>
  <label><input type="radio" name="payment" value="Cash"> Cash</label>
</form>
```

```
<script>
let payments = document.getElementsByName("payment");
payments.forEach(p => {
  p.addEventListener("change", () => {
    alert("You selected: " + p.value);
  });
});
</script> */

// --------------------------------------
// Chapter # 85 [Form validation: ZIP codes]
// --------------------------------------

// 1 [Basic ZIP code length check]
/* <input type="text" id="zip" placeholder="Enter ZIP code">
<button onclick="checkZip()">Validate</button>

<script>
function checkZip() {
  let zip = document.getElementById("zip").value;
  if (zip.length === 5) {
    alert("Valid ZIP code ✅");
  } else {
    alert("ZIP must be 5 digits ❌");
  }
```

```
}
</script> */

// 2 [Check only digits (numeric)]
/* <input type="text" id="zip2" placeholder="ZIP code (only numbers)">
<button onclick="validateZipNum()">Check</button>

<script>
function validateZipNum() {
  let zip = document.getElementById("zip2").value;
  if (/^\d+$/.test(zip)) {
    alert("Only numbers entered ✅");
  } else {
    alert("ZIP code must contain only digits ❌");
  }
}
</script> */

// 3 [Combine digits + length (5 digit ZIP)]
/* <input type="text" id="zip3" placeholder="Enter ZIP (5 digits)">
<button onclick="checkZip5()">Check</button>

<script>
function checkZip5() {
  let zip = document.getElementById("zip3").value;
  if (/^\d{5}$/.test(zip)) {
    alert("Valid ZIP ✅");
  } else {
    alert("ZIP must be exactly 5 digits ❌");
  }
}
</script> */

// 4 [ZIP + 4 format (like 12345-6789)]
/* <input type="text" id="zip4" placeholder="12345-6789">
<button onclick="validateZipPlus4()">Check</button>

<script>
function validateZipPlus4() {
  let zip = document.getElementById("zip4").value;
  if (/^\d{5}(-\d{4})?$/.test(zip)) {
    alert("Valid ZIP+4 ✅");
  } else {
    alert("Format must be 12345 or 12345-6789 ❌");
  }
}
</script> */

// 5 [Country-specific ZIP check (example: Pakistan)]
/* <input type="text" id="pkzip" placeholder="PK ZIP (5 digits)">
<button onclick="validatePkZip()">Check</button>
```

```
<script>

function validatePkZip() {

  let zip = document.getElementById("pkzip").value;

  if (/^\d{5}$/.test(zip)) {

    alert("Valid Pakistan ZIP ✅");

  } else {

    alert("Pakistani ZIP must be 5 digits ❌");

  }

}

</script> */


// ----------------------------------------

// Chapter # 86 [Form validation: email]

// ----------------------------------------


// 1 [Basic check (@ hona chahiye)]

/* <input type="text" id="email1" placeholder="Enter email">

<button onclick="checkEmail1()">Validate</button>


<script>

function checkEmail1() {

  let email = document.getElementById("email1").value;

  if (email.includes("@")) {

    alert("Valid email ✅");

  } else {

    alert("Email must contain '@' ❌");
  }
}
</script */

// 2 [Basic regex check (username@domain)]
/* <input type="text" id="email2" placeholder="Enter email">
<button onclick="checkEmail2()">Validate</button>

<script>
function checkEmail2() {
  let email = document.getElementById("email2").value;
  let pattern = /^\S+@\S+\.\S+$/;
  if (pattern.test(email)) {
    alert("Valid email ✅");
  } else {
    alert("Invalid email ❌");
  }
}
</script> */

// 3 [Advance regex (standard email format)]
/* <input type="text" id="email3" placeholder="Enter email">
<button onclick="checkEmail3()">Validate</button>

<script>
function checkEmail3() {
  let email = document.getElementById("email3").value;
  let pattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}$/;
  if (pattern.test(email)) {
    alert("Valid email ✅");
  } else {
    alert("Invalid email ❌");
  }
}
</script> */

// 4 [HTML5 built-in validation]
/* <form>
  <input type="email" id="email4" required>
  <button type="submit">Submit</button>
</form> */

// 5 [Prevent invalid email before submit]
/* <form onsubmit="return validateForm()">
  <input type="text" id="email5" placeholder="Enter email">
  <button type="submit">Submit</button>
</form>

<script>
function validateForm() {
  let email = document.getElementById("email5").value;
  let pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!pattern.test(email)) {
    alert("Please enter a valid email ❌");
    return false; // prevent submit
  }
  return true;
}
</script> */

// ---------------------------------------
// Chapter # 87 [Exceptions: try and catch]
// ---------------------------------------

// 1 [Syntax (basic)]
// try {
//   // code that may throw error
// } catch (error) {
//   // error handle karne ka code
// } finally {
//   // (optional) hamesha run hota hai
// }

// 2 [Basic error handling]
// try {
//   let x = y + 5; // y defined nahi hai
// } catch (err) {
//   console.log("Error: " + err.message);
// }

// 3 [Multiple safe code blocks]
// try {
//   console.log("Start");
//   let num = 10 / 0;
//   console.log(num);
// } catch (err) {
//   console.log("Something went wrong!");
// }

// 4 [finally block (always run)]
// try {
//   let data = JSON.parse("{name: 'Ali'}"); // invalid JSON
// } catch (err) {
//   console.log("Invalid JSON!");
// } finally {
//   console.log("Parsing attempt finished.");
// }

// 5 [Custom error throw + catch]
// try {
//   let age = -5;
//   if (age < 0) throw new Error("Age cannot be negative!");
// } catch (err) {
//   console.log("Caught: " + err.message);
// }

// 6 [Divide by zero check]
// function divide(a, b) {
//   try {
//     if (b === 0) throw new Error("Cannot divide by zero!");
//     return a / b;
//   } catch (err) {
//     return err.message;
//   }
// }
// console.log(divide(10, 0));

// 7 [JSON validation]
// try {
//   let json = '{"name":"Hasnain","age":22}';
//   let user = JSON.parse(json);
//   console.log(user.name);
// } catch (err) {
//   console.log("Invalid JSON data");
// }

// 8 [Nested try...catch]
// try {
//   try {
//     let arr = null;
//     console.log(arr.length); // error
//   } catch (innerErr) {
//     console.log("Inner error: " + innerErr.message);
//   }
// } catch (outerErr) {
//   console.log("Outer error: " + outerErr.message);
// }

// 9 [Function error handling]
// function getUser(obj) {
//   try {
//     return obj.name.toUpperCase();
//   } catch (err) {
//     return "Invalid user object!";
//   }
// }
// console.log(getUser({}));

// 10 [Async code with try/catch (Promise)]
// async function fetchData() {
//   try {
//     let res = await fetch("https://invalid-url");
//     let data = await res.json();
//     console.log(data);
//   } catch (err) {
//     console.log("Fetch failed: " + err.message);
//   }
// }
// fetchData();

// 11 [Typical real-world form validation]
// function validateEmail(email) {
//   try {
//     if (!email.includes("@")) throw new Error("Email must contain @");
//     if (!email.includes(".")) throw new Error("Email must contain .");
//     return "Valid email ✅";
//   } catch (err) {
//     return "Error: " + err.message;
//   }
// }
// console.log(validateEmail("testgmailcom"));

// --------------------------------------
// Chapter # 88 [Exceptions: throw]
// --------------------------------------

// Syntax
// throw expression;

// 1 [Throw string]
// try {
//   throw "Something went wrong!";
// } catch (err) {
//   console.log("Error: " + err);
// }

// 2 [Throw number]
// try {
//   throw 404;
// } catch (err) {
//   console.log("Error Code: " + err);
// }

// 3 [Throw object]
// try {
//   throw { message: "Invalid Input", code: 400 };
// } catch (err) {
//   console.log(err.code + ": " + err.message);
// }

// 4 [Throw new Error()]
// try {
//   throw new Error("Custom error occurred!");
// } catch (err) {
//   console.log(err.name + ": " + err.message);
// }

// 5 [Throw in function]
// function divide(a, b) {
//   if (b === 0) {
//     throw new Error("Cannot divide by zero!");
//   }
//   return a / b;
// }

// try {
//   console.log(divide(10, 0));
// } catch (err) {
//   console.log(err.message);
// }

// 6 [Nested throw/catch]
// try {
//   try {
//     throw new Error("Inner error");
//   } catch (e) {
//     console.log("Caught inside: " + e.message);
//     throw e; // re-throw
//   }
// } catch (e) {
//   console.log("Caught outside: " + e.message);
// }

// 7 [Throw in JSON parsing]
// function parseJSON(str) {
//   try {
//     return JSON.parse(str);
//   } catch {
//     throw new Error("Invalid JSON format!");
//   }
// }

// try {
//   parseJSON("{name:'Ali'}");
// } catch (err) {
//   console.log(err.message);
// }

// 8 [Throw in async function]
// async function getData() {
//   let ok = false;
//   if (!ok) throw new Error("Data not available!");
//   return "Data found";
// }

// getData()
//   .then(res => console.log(res))
//   .catch(err => console.log("Caught: " + err.message));

// 9 [Typical: Email validation with throw]
// function validateEmail(email) {
//   if (!email.includes("@")) throw new Error("Email must contain @");
//   if (!email.includes(".")) throw new Error("Email must contain .");
//   return "Valid email ✅";
// }

// try {
//   console.log(validateEmail("testgmailcom"));
// } catch (err) {
//   console.log("Error: " + err.message);
// }

// --------------------------------------
// Chapter # 89 [Handling events within JavaScript]
// --------------------------------------

// Syntax
// element.addEventListener("eventName", function);
/* <button onclick="myFunc()">Click me</button> */

// 1 [Button click event]
/* <button id="btn">Click me</button>
<script>
document.getElementById("btn").addEventListener("click", () => {
  alert("Button clicked!");
});
```

```
</script> */


// 2 [Mouse over event]
/* <div id="box" style="width:100px;height:100px;background:lightblue"></div>
<script>
document.getElementById("box").addEventListener("mouseover", () => {
  console.log("Mouse entered the box!");
});
</script> */


// 3 [Mouse out event]
/* <div id="box2" style="width:100px;height:100px;background:lightgreen"></div>
<script>
document.getElementById("box2").addEventListener("mouseout", () => {
  console.log("Mouse left the box!");
});
</script> */


// 4 [Input field ke andar typing (keyup)]
/* <input type="text" id="name" placeholder="Type something...">
<script>
document.getElementById("name").addEventListener("keyup", (e) => {
  console.log("You typed: " + e.target.value);
});
</script> */
```

```
// 5 [Form submit event]

/* <form id="myForm">

  <input type="text" required>

  <button type="submit">Submit</button>

</form>

<script>

document.getElementById("myForm").addEventListener("submit", (e) => {

  e.preventDefault(); // page reload stop

  console.log("Form submitted!");

});

</script> */


// 6 [Double click event]

/* <p id="para">Double click me!</p>

<script>

document.getElementById("para").addEventListener("dblclick", () => {

  alert("You double clicked!");

});

</script> */


// 7 [Keyboard event (Enter key)]

/* <input type="text" id="inp" placeholder="Press Enter">

<script>

document.getElementById("inp").addEventListener("keydown", (e) => {

  if (e.key === "Enter") {

    alert("Enter pressed!");
  }
});
</script> */

// 8 [Window resize event]
/* <script>
window.addEventListener("resize", () => {
  console.log("Window resized: " + window.innerWidth + "px");
});
</script> */

// 9 [Change event on dropdown]
/* <select id="city">
  <option value="">Select City</option>
  <option value="Karachi">Karachi</option>
  <option value="Lahore">Lahore</option>
</select>
<script>
document.getElementById("city").addEventListener("change", (e) => {
  console.log("You selected: " + e.target.value);
});
</script> */

// 10 [Event delegation (parent ke through child handle)]
/* <ul id="list">
  <li>Apple</li>
  <li>Mango</li>

  <li>Banana</li>

</ul>

<script>

document.getElementById("list").addEventListener("click", (e) => {

  if (e.target.tagName === "LI") {

    alert("You clicked: " + e.target.innerText);

  }

});

</script> */


// 🖱 Mouse Events

// 1. click

// element par mouse click hone par fire hota hai.

// 👉 Example: button dabana.


// 2. dblclick

// mouse double click par.

// 👉 Example: text edit mode open.


// 3. mouseover

// mouse pointer element ke upar aate hi.

// 👉 Example: hover effect ya tooltip show.


// 4. mouseout

// mouse pointer element se bahar nikalte hi.
```

// 👉 Example: hover effect hatana, tooltip hide.

// 5. contextmenu

// right-click hone par.

// 👉 Example: custom context menu show.

// ⌨️ Keyboard Events

// 6. keydown

// key press hone ke turant baad fire hota hai.

// 👉 Example: shortcuts, Enter detect karna.

// 7. keyup

// jab key release hoti hai.

// 👉 Example: typing ke baad live validation.

// 8. keypress (old, ab kam use hota hai)

// jab printable key press hoti hai.

// 👉 Example: character logging.

// 📝 Form Events

// 9. input

// text field ki value har character par update hoti hai.

// 👉 Example: live search, password strength.

// 10. change

// input ki value tab trigger hoti hai jab focus lose hone ke baad change confirm ho jaye.

// 👉 Example: dropdown, checkbox, radio buttons.

// 11. submit

// form submit hone par.

// 👉 Example: validation aur data send karna.

// 12. focus

// jab input field active ho jaye (cursor andar ho).

// 👉 Example: highlight ya help text show karna.

// 13. blur

// jab input field se cursor bahar nikal jaye.

// 👉 Example: field validate karna.

// ▯ Window/Document Events

// 14. resize

// jab window ka size change hota hai.

// 👉 Example: responsive design adjust karna.

// 15. scroll

// jab user page scroll kare.

// 👉 Example: sticky navbar, infinite scroll, animations.