**🛠 Roadmap---{JS}**

**🧱 JavaScript Level Breakdown**

**🟢 Beginner Level (Basic foundation)---DONE**

Yaha tu language ka structure aur base samajhta hai  
**Topics:**

* Variables (let, const, var)
* Data types (string, number, boolean, array, object)
* Operators (+, -, ===, etc.)
* Conditional statements (if, else, switch)
* Loops (for, while, for...of, for...in)
* Functions (normal + arrow functions)
* Arrays (push, pop, slice, etc.)
* Objects (key-value pairs)
* Basic DOM manipulation (getElementById, innerHTML)

**🟡 Intermediate Level (Logic + Async + Real coding)---DONE**

Ye level par tu **browser ke behavior aur timing** samajhta hai  
**Topics:**

* setTimeout() & setInterval() (asynchronous timing)
* Callback functions
* Promises (resolve, reject, .then(), .catch())
* async/await (modern async handling)
* Error handling with try...catch
* ES6 features (spread, destructuring, template literals)
* Event handling (click, input, etc.)
* Fetch API / AJAX (data lana server se)
* LocalStorage / SessionStorage
* Basic debugging (console.log, breakpoints)

**🔵 Advanced Level (Pro coder zone)**

Ye level pe tu JS se bade projects banata hai  
**Topics:**

* Closures, hoisting, scope, prototype chain
* Event loop & microtasks
* Advanced async patterns (Promise.all, race)
* Modules (import/export)
* OOP in JS (classes, inheritance)
* DOM optimization, debouncing, throttling
* Frontend frameworks (React, Vue)
* Tooling (Webpack, Babel, NPM)
* API integration with error handling
* Testing (Jest, Mocha)
* Performance optimization

🧠 **In short:**

| **Level** | **Skills** | **Example** |
| --- | --- | --- |
| Beginner | Syntax, loops, arrays, DOM basics | “Hello World” page |
| Intermediate | Async code, Promises, logic building | API fetch, timers |
| Advanced | Frameworks, performance, OOP | React apps, large JS projects |

# **🌟 Top 50 Basic JavaScript Interview Questions + Short Answers 🌟**

**🟢 Section 1: Basics & Syntax (1–10)**

1. **What is JavaScript?**

Programming language for interactive web pages.

1. **Difference between JS and Java?**

JS runs in browser, Java is standalone; syntax different.

1. **How to include JS in HTML?**

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

1. **Data types in JS?**

String, Number, Boolean, Undefined, Null, Symbol, BigInt, Object

1. **Difference between var, let, const?**

var – function scope  
let – block scope  
const – block scope + cannot reassign

1. **What is NaN?**

Not a Number, e.g. "abc"/2

1. **Difference undefined vs null?**

undefined – value not assigned  
null – explicitly empty

1. **What is Hoisting?**

Variable/function declaration move to top during execution

1. **What is strict mode?**

"use strict";

Enforces better coding, errors thrown for bad practices

1. **What are comments in JS?**

// single line

/\* multi-line \*/

**🟡 Section 2: Variables & Operators (11–20)**

1. Difference between == and ===?

== – value check, type convert  
=== – value + type check

1. What is operator precedence?

Determines order in which operations are performed

1. Difference between postfix i++ and prefix ++i?

i++ – returns old value  
++i – returns new value

1. What is ternary operator?

let a = 5;

let result = a > 3 ? "Yes" : "No";

1. What is typeof operator?

typeof 123; // "number"

1. Difference between + and +=?

+ addition, += addition + assignment

1. What is NaN check?

isNaN("abc"); // true

1. Difference between Infinity and -Infinity?

Positive and negative infinite numbers

1. Can variables be named with numbers?

Yes, but cannot start with a number

1. What is template literal?

let name = "Trupti";

console.log(`Hello ${name}`);

**🔵 Section 3: Functions (21–30)**

1. What is a function?

Block of code that performs a task

1. Function declaration vs expression?

function a(){} // declaration

let b = function(){}; // expression

1. What is arrow function?

let add = (x,y) => x+y;

1. Can functions return values?

Yes, using return

1. What is callback function?

Function passed as argument and called later

1. Difference between arguments and parameters?

Parameter – variable in function definition  
Argument – actual value passed

1. Can functions be stored in variables?

Yes, functions are first-class citizens

1. What is IIFE? (Immediately Invoked Function Expression)

(function(){ console.log("Hi"); })();

1. Difference between normal function & arrow?

Arrow has no this, cannot be used as constructor

1. What is rest parameter?

function sum(...nums){ return nums.reduce((a,b)=>a+b); }

**🟢 Section 4: Arrays & Objects (31–40)**

1. How to create an array?

let arr = [1,2,3];

1. Access array element?

console.log(arr[0]); // 1

1. Add element to array?

arr.push(4); // [1,2,3,4]

1. Remove element from array?

arr.pop(); // removes last

1. Loop through array?

arr.forEach(item => console.log(item));

1. Difference between array and object?

Array – ordered list  
Object – key-value pairs

1. How to access object property?

let obj = {name:"Trupti"};

console.log(obj.name); // Trupti

1. Add property to object?

obj.age = 22;

1. Delete property from object?

delete obj.age;

1. Object.keys and Object.values?

Object.keys(obj); // ["name"]

Object.values(obj); // ["Trupti"]

**🟡 Section 5: DOM & Events (41–45)**

1. What is DOM?

Document Object Model – JS access HTML elements

1. Get element by ID?

document.getElementById("myId");

1. Event listener example?

document.getElementById("btn").addEventListener("click", () => alert("Clicked!"));

1. Change innerHTML?

document.getElementById("demo").innerHTML = "Hello!";

1. Difference between onClick attribute & addEventListener?

onClick – inline, single handler  
addEventListener – multiple handlers allowed

**🔵 Section 6: Async & Promises (46–50)**

1. What is asynchronous JS?

Code that runs **later** without blocking execution

1. What is a promise?

Object representing **future value**

let p = new Promise(res=>res("Done"));

1. How to use .then()?

p.then(msg => console.log(msg));

1. What is async/await?

Modern way to handle promises, write code **synchronous style**

async function f(){ let res = await p; console.log(res);}

1. How to handle promise error?

p.catch(err => console.log(err));