100+ JavaScript Real-World Problems (With Solutions)

1. Basics & Strings

Q1. Reverse a string

```
function reverseString(str) {
  return str.split("").reverse().join("");
}
console.log(reverseString("hello")); // "olleh"
```

Q2. Check if a string is a palindrome

```
function isPalindrome(str) {
  const rev = str.split("").reverse().join("");
  return str === rev;
}
console.log(isPalindrome("madam")); // true
```

Q3. Count vowels in a string

```
function countVowels(str) {
  return str.match(/[aeiou]/gi)?.length || 0;
}
console.log(countVowels("javascript")); // 3
```

Q4. Find the first non-repeating character

```
function firstNonRepeatingChar(str) {
  for (let char of str) {
    if (str.indexOf(char) === str.lastIndexOf(char)) return char;
  }
  return null;
}
console.log(firstNonRepeatingChar("swiss")); // "w"
```

2. Numbers

Q5. Check if a number is prime

```
function isPrime(n) {
  if (n < 2) return false;
  for (let i = 2; i <= Math.sqrt(n); i++) {
    if (n % i === 0) return false;
  }
  return true;
}
console.log(isPrime(7)); // true
```

Q6. Find factorial of a number

```
function factorial(n) {
  return n === 0 ? 1 : n * factorial(n - 1);
}
console.log(factorial(5)); // 120
```

Q7. Find nth Fibonacci number

```
function fibonacci(n) {
  if (n <= 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}
console.log(fibonacci(6)); // 8
```

Q8. Find greatest common divisor (GCD)

```
function gcd(a, b) {
  return b === 0 ? a : gcd(b, a % b);
}
console.log(gcd(48, 18)); // 6
```

3. Arrays

Q9. Find the maximum number in an array

```
const arr = [3, 7, 2, 9, 4];
console.log(Math.max(...arr)); // 9
```

Q10. Remove duplicates from an array

```
const unique = [...new Set([1, 2, 2, 3, 4, 4])];
console.log(unique); // [1,2,3,4]
```

Q11. Flatten a nested array

```
function flatten(arr) {
  return arr.flat(Infinity);
}
console.log(flatten([1, [2, [3]]])); // [1,2,3]
```

Q12. Find intersection of two arrays

```
function intersection(a, b) {
  return a.filter(x => b.includes(x));
}
console.log(intersection([1,2,3], [2,3,4])); // [2,3]
```

4. Objects

Q13. Clone an object

```
const obj = {a:1, b:{c:2}};
const clone = JSON.parse(JSON.stringify(obj));
console.log(clone);
```

Q14. Count character frequency

```
function charFrequency(str) {
  return [...str].reduce((acc, char) => {
    acc[char] = (acc[char] || 0) + 1;
    return acc;
  }, {});
}
console.log(charFrequency("banana"));
// {b:1, a:3, n:2}
```

5. Asynchronous JS

Q15. Fetch data from API

```
async function fetchData() {
  const res = await fetch("https://jsonplaceholder.typicode.com/posts/1");
  const data = await res.json();
  console.log(data);
}
fetchData();
```

Q16. Delay execution using Promise

```
function delay(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
delay(1000).then(() => console.log("Executed after 1s"));
```

6. DOM

Q17. Change text on button click

```
<button id="btn">Click Me</button>
<script>
document.getElementById("btn").addEventListener("click", () => {
  document.getElementById("btn").innerText = "Clicked!";
});
</script>
```

Q18. Toggle dark mode

```
document.body.classList.toggle("dark-mode");
```

7. Real-World Utilities

Q19. Debounce function

```
function debounce(fn, delay) {
  let timer;
  return function(...args) {
    clearTimeout(timer);
    timer = setTimeout(() => fn.apply(this, args), delay);
  };
}
```

Q20. Throttle function

```
function throttle(fn, limit) {
  let waiting = false;
  return function(...args) {
    if (!waiting) {
      fn.apply(this, args);
      waiting = true;
      setTimeout(() => waiting = false, limit);
    }
  };
}
```

Q21. Shuffle an array

```
function shuffle(arr) {
  return arr.sort(() => Math.random() - 0.5);
}
console.log(shuffle([1,2,3,4,5]));
```

Q22. Generate random hex color

```
function randomColor() {
  return "#" + Math.floor(Math.random()*16777215).toString(16);
}
console.log(randomColor());
```

8. Arrays

Q23. Find the second largest number in an array

```
function secondLargest(arr) {
  let unique = [...new Set(arr)];
  unique.sort((a, b) => b - a);
  return unique[1];
}
console.log(secondLargest([10, 5, 8, 20, 20])); // 10
```

Q24. Rotate an array by k positions

```
function rotateArray(arr, k) {
  k = k % arr.length;
  return arr.slice(-k).concat(arr.slice(0, -k));
}
console.log(rotateArray([1,2,3,4,5], 2)); // [4,5,1,2,3]
```

Q25. Find missing number in a sequence

```
function missingNumber(arr) {
  const n = arr.length + 1;
  const total = (n * (n+1)) / 2;
  return total - arr.reduce((a,b) => a+b, 0);
}
console.log(missingNumber([1,2,4,5])); // 3
```

Q26. Merge two sorted arrays

```
function mergeSorted(a, b) {
  return [...a, ...b].sort((x, y) => x - y);
}
console.log(mergeSorted([1,3,5], [2,4,6])); // [1,2,3,4,5,6]
```

Q27. Find duplicates in an array

```
function findDuplicates(arr) {
  return arr.filter((item, index) => arr.indexOf(item) !== index);
}
console.log(findDuplicates([1,2,3,2,4,1])); // [2,1]
```

Q28. Chunk an array into smaller arrays

```
function chunk(arr, size) {
  let res = [];
  for (let i = 0; i < arr.length; i += size) {
    res.push(arr.slice(i, i + size));
  }
  return res;
}
console.log(chunk([1,2,3,4,5], 2)); // [[1,2],[3,4],[5]]
```

Q29. Find the longest word in an array

```
function longestWord(arr) {
  return arr.reduce((a,b) => a.length > b.length ? a : b);
}
console.log(longestWord(["js", "javascript", "python"])); // "javascript"
```

Q30. Count occurrences of elements in array

```
function countOccurrences(arr) {
  return arr.reduce((acc, val) => {
    acc[val] = (acc[val] || 0) + 1;
    return acc;
  }, {});
}
console.log(countOccurrences([1,2,2,3,3,3]));
// {1:1, 2:2, 3:3}
```

Q31. Remove falsy values from an array

```
function removeFalsy(arr) {
  return arr.filter(Boolean);
}
console.log(removeFalsy([0, 1, false, 2, "", 3])); // [1,2,3]
```

Q32. Find union of two arrays

```
function union(a, b) {
  return [...new Set([...a, ...b])];
}
console.log(union([1,2], [2,3])); // [1,2,3]
```

Q33. Find difference of two arrays

```
function difference(a, b) {
  return a.filter(x => !b.includes(x));
}
console.log(difference([1,2,3], [2,3,4])); // [1]
```

9. Objects

Q34. Convert object to array of key-value pairs

```
const obj = {a:1, b:2};
console.log(Object.entries(obj)); // [["a",1],["b",2]]
```

Q35. Convert array of pairs to object

```
const arr = [["a",1],["b",2]];
console.log(Object.fromEntries(arr)); // {a:1, b:2}
```

Q36. Deep clone an object

```
function deepClone(obj) {
  return structuredClone(obj);
}
console.log(deepClone({x:1, y:{z:2}}));
```

Q37. Merge two objects

```
const obj1 = {a:1, b:2};
const obj2 = {b:3, c:4};
const merged = {...obj1, ...obj2};
console.log(merged); // {a:1, b:3, c:4}
```

Q38. Check if object is empty

```
function isEmpty(obj) {
  return Object.keys(obj).length === 0;
}
console.log(isEmpty({})); // true
```

Q39. Sort array of objects by key

```
const users = [{name:"John", age:25},{name:"Alice", age:20}];
users.sort((a,b) => a.age - b.age);
console.log(users);
```

Q40. Get nested property safely (optional chaining)

```
const user = {profile:{email:"test@mail.com"}};
console.log(user?.profile?.email); // "test@mail.com"
```

10. ES6+ Features

Q41. Default function parameters

```
function greet(name="Guest") {
  return `Hello ${name}`;
}
console.log(greet()); // Hello Guest
```

Q42. Rest parameter example

```
function sum(...nums) {
  return nums.reduce((a,b)=>a+b,0);
}
console.log(sum(1,2,3)); // 6
```

Q43. Destructure with defaults

```
const {x=10, y=20} = {x:5};
console.log(x, y); // 5 20
```

Q44. Template literals

```
let name="Krish";
console.log(`Welcome, ${name}!`);
```

Q45. Use of Set for unique values

```
const unique = [...new Set([1,2,2,3,3,4])];
console.log(unique); // [1,2,3,4]
```

11. Asynchronous JavaScript

Q46. Create a function that resolves after 2 seconds

```
function waitTwoSeconds() {
  return new Promise(resolve => {
    setTimeout(() => resolve("Done!"), 2000);
  });
}
waitTwoSeconds().then(console.log); // Done! (after 2s)
```

Q47. Async/Await example with error handling

```
async function fetchData() {
  try {
    let res = await fetch("https://jsonplaceholder.typicode.com/posts/1");
    let data = await res.json();
    console.log(data);
  } catch (err) {
    console.error("Error:", err);
  }
}
fetchData();
```

Q48. Run promises in parallel

```
Promise.all([
  fetch("https://jsonplaceholder.typicode.com/posts/1"),
  fetch("https://jsonplaceholder.typicode.com/posts/2")
]).then(responses => console.log("All fetched!"));
```

Q49. Run promises in race

```
Promise.race([
  new Promise(r => setTimeout(()=>r("Fast"),500)),
  new Promise(r => setTimeout(()=>r("Slow"),2000))
]).then(console.log); // "Fast"
```

Q50. Retry a function until success

```
async function retry(fn, retries=3) {
  for (let i=0; i<retries; i++) {
    try { return await fn(); }
    catch (e) { if (i === retries-1) throw e; }
  }
}
```

Q51. Sequential promise execution

```
async function runSequential(tasks) {
  for (let t of tasks) {
    console.log(await t());
  }
}
runSequential([
  ()=>Promise.resolve("First"),
  ()=>Promise.resolve("Second")
]);
```

Q52. Timeout promise

```
function timeoutPromise(ms) {
  return new Promise((_, reject) =>
    setTimeout(()=>reject("Timeout!"), ms)
  );
}
timeoutPromise(1000).catch(console.error);
```

Q53. Polling with setInterval

```
function poll(fn, interval) {
  const id = setInterval(async () => {
    let result = await fn();
    if (result) clearInterval(id);
  }, interval);
}
```

Q54. Event loop demo (micro vs macro task)

```
console.log("Start");
setTimeout(()=>console.log("Timeout"),0);
Promise.resolve().then(()=>console.log("Promise"));
console.log("End");
// Order: Start → End → Promise → Timeout
```

Q55. Implement sleep function

```
const sleep = ms => new Promise(res => setTimeout(res, ms));
(async () => {
  console.log("Wait...");
  await sleep(1000);
  console.log("Done!");
})();
```

12. DOM Manipulations

Q56. Change background color on button click

```
<button onclick="document.body.style.background='lightblue'">Change</button>
```

Q57. Toggle visibility of an element

```
function toggle(el) {
  el.style.display = (el.style.display === "none") ? "block" : "none";
}
```

Q58. Create and append a new element

```
const div = document.createElement("div");
div.textContent = "Hello DOM!";
document.body.appendChild(div);
```

Q59. Remove an element from DOM

```
document.getElementById("myEl").remove();
```

Q60. Handle form submit without reload

```
document.querySelector("form").addEventListener("submit", e => {
  e.preventDefault();
  console.log("Form submitted!");
});
```

Q61. Image lazy loading

```
<img src="placeholder.jpg" data-src="real.jpg" class="lazy">
<script>
document.querySelectorAll(".lazy").forEach(img=>{
  img.src = img.dataset.src;
});
</script>
```

Q62. Add class to element

```
document.querySelector("#box").classList.add("active");
```

Q63. Get value of input field

```
const val = document.querySelector("#input").value;
```

Q64. Smooth scroll to section

```
document.querySelector("#section").scrollIntoView({behavior:"smooth"});
```

Q65. Copy text to clipboard

```
navigator.clipboard.writeText("Hello World").then(()=>console.log("Copied!"));
```

13. Real-World Utilities

Q66. Capitalize first letter of each word

```
function capitalize(str) {
  return str.replace(/\b\w/g, c => c.toUpperCase());
}
console.log(capitalize("hello world")); // Hello World
```

Q67. Generate UUID

```
function uuid() {
  return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, c => {
    const r = Math.random()*16|0;
    const v = c === 'x' ? r : (r&0x3|0x8);
    return v.toString(16);
  });
}
console.log(uuid());
```

Q68. Check if email is valid

```
function isValidEmail(email) {
  return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);
}
console.log(isValidEmail("test@mail.com")); // true
```

Q69. Convert RGB to Hex

```
function rgbToHex(r,g,b) {
  return "#" + [r,g,b].map(x=>x.toString(16).padStart(2,"0")).join("");
}
console.log(rgbToHex(255,99,71)); // #ff6347
```

Q70. Convert Hex to RGB

```
function hexToRgb(hex) {
  let num = parseInt(hex.slice(1),16);
  return {
    r: (num >> 16) & 255,
    g: (num >> 8) & 255,
    b: num & 255
  };
}
console.log(hexToRgb("#ff6347")); // {r:255,g:99,b:71}
```

14. OOP in JavaScript

Q71. Create a class with constructor and method

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  greet() {
    return `Hello, my name is ${this.name}`;
  }
}
const p = new Person("Krish", 25);
console.log(p.greet());
```

Q72. Inheritance example

```
class Employee extends Person {
  constructor(name, age, salary) {
    super(name, age);
    this.salary = salary;
  }
  getSalary() {
    return this.salary;
  }
}
const e = new Employee("Alice", 28, 5000);
console.log(e.getSalary()); // 5000
```

Q73. Static method in class

```
class MathUtil {
  static square(x) { return x*x; }
}
console.log(MathUtil.square(5)); // 25
```

Q74. Private fields

```
class BankAccount {
  #balance = 0;
  deposit(amount) { this.#balance += amount; }
  getBalance() { return this.#balance; }
}
const acc = new BankAccount();
acc.deposit(100);
console.log(acc.getBalance()); // 100
```

Q75. Getter and Setter

```
class Person {
  constructor(name) { this._name = name; }
  get name() { return this._name; }
  set name(n) { this._name = n; }
}
```

Q76. Polymorphism example

```
class Animal {
  speak() { console.log("Animal speaks"); }
}
class Dog extends Animal {
  speak() { console.log("Dog barks"); }
}
new Dog().speak(); // Dog barks
```

Q77. Prototype method example

```
function Car(model) { this.model = model; }
Car.prototype.drive = function() { console.log(`${this.model} is driving`); }
const c = new Car("Tesla");
c.drive();
```

Q78. Check instance of a class

```
console.log(c instanceof Car); // true
```

Q79. Mixins example

```
let mixin = { greet() { console.log("Hello!"); } };
class User {}
Object.assign(User.prototype, mixin);
new User().greet();
```

Q80. Object.create inheritance

```
const parent = { greet() { console.log("Hi!"); } };
const child = Object.create(parent);
child.greet(); // Hi!
```

15. ES6+ Advanced

Q81. Destructure nested objects

```
const obj = { a:1, b:{ c:2 } };
const { b:{c} } = obj;
console.log(c); // 2
```

Q82. Array destructuring with rest

```
const [first, ...rest] = [1,2,3,4];
console.log(rest); // [2,3,4]
```

Q83. Optional chaining

```
const user = {};
console.log(user?.profile?.email); // undefined
```

Q84. Nullish coalescing

```
const val = null ?? "default";
console.log(val); // "default"
```

Q85. Dynamic property keys

```
const key = "name";
const obj = { [key]: "Krish" };
console.log(obj.name); // Krish
```

Q86. Tagged template literals

```
function tag(strings, name) {
  return `${strings[0]}${name.toUpperCase()}`;
}
console.log(tag`Hello ${"krish"}`); // Hello KRISH
```

Q87. Symbols for unique keys

```
const sym = Symbol("id");
const obj = { [sym]: 123 };
console.log(obj[sym]); // 123
```

Q88. Generator function example

```
function* gen() { yield 1; yield 2; }
const g = gen();
console.log(g.next().value); // 1
```

16. Performance & Utilities

Q89. Debounce function

```
function debounce(fn, delay) {
  let timer;
  return function(...args) {
    clearTimeout(timer);
    timer = setTimeout(() => fn.apply(this, args), delay);
  }
}
```

Q90. Throttle function

```
function throttle(fn, limit) {
  let waiting = false;
  return function(...args) {
    if(!waiting) {
      fn.apply(this, args);
      waiting = true;
      setTimeout(()=>waiting=false, limit);
    }
  }
}
```

Q91. Memoization

```
function memo(fn) {
  const cache = {};
  return (...args) => {
    const key = args.toString();
    if(cache[key]) return cache[key];
    return cache[key] = fn(...args);
  }
}
```

Q92. Lazy loading image

```
const imgs = document.querySelectorAll("img[data-src]");
imgs.forEach(img => img.src = img.dataset.src);
```

Q93. Throttle scroll event

```
window.addEventListener("scroll", throttle(()=>console.log("Scroll!"),200));
```

Q94. Deep clone object

```
const clone = structuredClone(obj);
```

17. Security & Validation

Q95. Validate URL

```
function isValidURL(url) {
  try { new URL(url); return true; } catch { return false; }
}
```

Q96. Escape HTML to prevent XSS

```
function escapeHTML(str) {
  return str.replace(/&/g,"&amp;").replace(/</g,"&lt;").replace(/>/g,"&gt;");
}
```

Q97. Check strong password

```
function isStrongPass(pw) {
  return /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@#$%^&+=]).{8,}$/.test(pw);
}
```

Q98. Sanitize input

```
function sanitize(str) { return str.replace(/<.*?>/g, ""); }
```

Q99. Generate random token

```
function token(len=16) {
  return [...Array(len)].map(()=>Math.random().toString(36)[2]).join("");
}
```

Q100. Check if code runs in browser

```
const isBrowser = typeof window !== "undefined";
console.log(isBrowser);
```

Q101. Remove duplicates from an array of objects based on a key

```
const users = [
  {id:1, name:"Alice"},
  {id:2, name:"Bob"},
  {id:1, name:"Alice"}
];

const uniqueUsers = [...new Map(users.map(u => [u.id, u])).values()];
console.log(uniqueUsers);
// [{id:1, name:"Alice"}, {id:2, name:"Bob"}]
```

Q102. Group array of objects by a key

```
const users = [
  {name:"Alice", role:"admin"},
  {name:"Bob", role:"user"},
  {name:"Charlie", role:"admin"}
];

const grouped = users.reduce((acc, u) => {
  (acc[u.role] = acc[u.role] || []).push(u);
  return acc;
}, {});
console.log(grouped);
/*
{
  admin: [{name:"Alice"}, {name:"Charlie"}],
  user: [{name:"Bob"}]
}
*/
```