Bootstrap Interview Questions & Answers (1-50)

1. What is Bootstrap?

Answer:
Bootstrap is a free, open-source front-end framework used for designing responsive and mobile-first websites using HTML, CSS, and JavaScript.

2. What are the key features of Bootstrap?

Answer:

- **Responsive Grid System**
- **Pre-designed Components** (Buttons, Modals, Alerts)
- **Flexbox Support**
- **Customization with SASS**
- **JavaScript Plugins**

3. What is the latest version of Bootstrap?

Answer:
The latest version is **Bootstrap 5** (as of 2024).

4. What is the Bootstrap Grid System?

Answer:
Bootstrap uses a **12-column responsive grid system** to create flexible layouts.
Example:

```
<div class="row">
  <div class="col-md-6">Column 1</div>
  <div class="col-md-6">Column 2</div>
</div>
```

5. What is the difference between Bootstrap 4 and Bootstrap 5?

Answer:

Feature	Bootstrap 4	Bootstrap 5
jQuery	Required	Removed
Grid System	Uses float	Uses flexbox
Forms	Basic	Advanced styling
Internet Explorer Support	Yes	No

6. How do you include Bootstrap in a project?

Answer:
Using **CDN**:

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

7. What are Bootstrap breakpoints?

Answer:
Bootstrap uses media query breakpoints:

Size	Prefix	Width
Extra Small	xs	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra Large	xl	≥1200px

8. How do you create a responsive image in Bootstrap?

Answer:
Use the .img-fluid class:

<img src="image.jpg" class="img-fluid" alt="Responsive Image">

9. What is the difference between .container and .container-fluid?

Answer:

- .container → Fixed width
- .container-fluid → Full width

10. How do you create a button in Bootstrap?

Answer:

<button class="btn btn-primary">Click Me</button>

11. What are Bootstrap button types?

Answer:

- **Primary (btn-primary)**
- **Secondary (btn-secondary)**
- **Success (btn-success)**
- **Danger (btn-danger)**
- **Warning (btn-warning)**

- **Info (btn-info)**
- **Light (btn-light)**
- **Dark (btn-dark)**

12. What is the difference between .d-none, .invisible, and display: none?

Answer:

- .d-none → Hides element and removes space
- .invisible → Hides element but keeps space
- display: none → Same as .d-none

13. How do you make a button full-width?

Answer:

<button class="btn btn-primary w-100">Full-Width Button</button>

14. What is Bootstrap's flexbox utility?

Answer:
Used for layout alignment.

Centered Content

15. How do you create a navigation bar in Bootstrap?

Answer:

<nav class="navbar navbar-expand-lg navbar-light bg-light">

Logo

</nav>

16. How do you make a sticky navbar?

Answer:

<nav class="navbar navbar-light bg-light fixed-top">

Sticky Navbar

</nav>

17. How do you add a dropdown menu in Bootstrap?

Answer:

<button class="btn btn-primary dropdown-toggle" data-bs-toggle="dropdown">Menu</button>

```
 <div class="dropdown-menu">
   <a class="dropdown-item" href="#">Item 1</a>
   <a class="dropdown-item" href="#">Item 2</a>
 </div>
</div>
```

18. How do you create a Bootstrap modal?

Answer:

```
<button data-bs-toggle="modal" data-bs-target="#myModal">Open Modal</button>
<div id="myModal" class="modal fade">
 <div class="modal-dialog">
   <div class="modal-content">Modal Content</div>
 </div>
</div>
```

19. What are Bootstrap cards?

Answer:
A flexible content container.

```
<div class="card">
 <div class="card-body">Card Content</div>
</div>
```

20. How do you create a carousel (slider) in Bootstrap?

Answer:

```
<div id="carouselExample" class="carousel slide" data-bs-ride="carousel">
 <div class="carousel-inner">
   <div class="carousel-item active">
     <img src="image1.jpg" class="d-block w-100">
   </div>
   <div class="carousel-item">
     <img src="image2.jpg" class="d-block w-100">
   </div>
 </div>
</div>
```

21. What is Bootstrap Toast?

Answer:
Used for showing messages/alerts.

```
<div class="toast show">
  <div class="toast-body">Hello!</div>
</div>
```

22. What is Bootstrap Breadcrumb?

Answer:
Shows navigation hierarchy.

```
<nav>
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active">Page</li>
  </ol>
</nav>
```

23. What is Bootstrap Tooltip?

Answer:
A hover text hint.

```
<button data-bs-toggle="tooltip" title="Tooltip text">Hover me</button>
```

24. How do you create a responsive table in Bootstrap?

Answer:

```
<table class="table table-responsive">
  <tr><td>Data</td></tr>
</table>
```

25. How do you add icons in Bootstrap?

Answer:
Using **Bootstrap Icons** (CDN required):

```
<i class="bi bi-alarm"></i>
```

26. What is Bootstrap Jumbotron?

Answer:
Jumbotron is a large padded container for highlighting content. It was removed in Bootstrap 5 but can be created using padding.

```
<div class="p-5 bg-light text-center">
```

```
  <h1>Welcome to My Website</h1>
</div>
```

27. How do you create a progress bar in Bootstrap?

Answer:

```
<div class="progress">
  <div class="progress-bar" style="width: 50%;">50%</div>
</div>
```

28. How do you make an element rounded in Bootstrap?

Answer:
Using .rounded classes:

```
<img src="image.jpg" class="rounded-circle">
```

29. How do you create a form in Bootstrap?

Answer:

```
<form>
  <div class="mb-3">
    <label class="form-label">Email:</label>
    <input type="email" class="form-control">
  </div>
</form>
```

30. How do you create an inline form?

Answer:

```
<form class="d-flex">
  <input type="text" class="form-control me-2">
  <button class="btn btn-primary">Submit</button>
</form>
```

31. What is Bootstrap Input Group?

Answer:
Used to group inputs and buttons.

```
<div class="input-group">
  <span class="input-group-text">@</span>
  <input type="text" class="form-control">
</div>
```

32. How do you create a floating label in Bootstrap?

Answer:

```
<div class="form-floating">
  <input type="email" class="form-control" placeholder="Email">
  <label>Email</label>
</div>
```

33. How do you align elements using Bootstrap?

Answer:
Using **Flexbox utilities** like .d-flex, .justify-content-*, and .align-items-*.

```
<div class="d-flex justify-content-center align-items-center">
  Centered Content
</div>
```

34. What is the use of .w-50, .h-100 in Bootstrap?

Answer:
Used for setting width and height.

- .w-50 → 50% width
- .h-100 → 100% height

35. How do you create a responsive sidebar in Bootstrap?

Answer:

```
<div class="d-flex flex-column flex-shrink-0 p-3 bg-light" style="width: 280px;">
  <a href="#" class="navbar-brand">Sidebar</a>
</div>
```

36. What is the difference between .row and .d-flex?

Answer:

- .row → Used in Bootstrap's **grid system**.
- .d-flex → Used for **flexbox layout**.

37. How do you create a responsive pricing table?

Answer:

```
<div class="card">
  <div class="card-header">Basic Plan</div>
  <div class="card-body">$9.99/month</div>
</div>
```

38. What is the use of text-truncate?

Answer:
Used for truncating long text.

This is a very long text that will be truncated.

39. How do you add a hover effect to a card?

Answer:

Hover Me

(Custom CSS needed for .hover-shadow)

40. How do you make a sticky footer?

Answer:

<footer class="fixed-bottom bg-dark text-white p-3 text-center">

Sticky Footer

</footer>

41. What is a Bootstrap Collapse?

Answer:
Used to show/hide content.

<button data-bs-toggle="collapse" data-bs-target="#demo">Toggle</button>

Collapsible Content

42. How do you create a vertical navbar?

Answer:

<nav class="nav flex-column">

Home

</nav>

43. How do you use Bootstrap Pills?

Answer:

<ul class="nav nav-pills">

<li class="nav-item">

Tab 1

44. How do you disable a button in Bootstrap?

Answer:

<button class="btn btn-primary disabled">Disabled</button>

45. What is the difference between .shadow-sm, .shadow, and .shadow-lg?

Answer:

- .shadow-sm → Small shadow
- .shadow → Default shadow
- .shadow-lg → Large shadow

46. How do you center text in Bootstrap?

Answer:
Using .text-center:

Centered Text

47. How do you create a full-width jumbotron-like section?

Answer:

Full-width Section

48. What is the difference between align-items and justify-content in Bootstrap?

Answer:

- align-items → Aligns items vertically
- justify-content → Aligns items horizontally

49. How do you create a sticky header?

Answer:

<header class="sticky-top bg-light p-3">

Sticky Header

</header>

50. What are Bootstrap utility classes?

Answer:
Utility classes are pre-defined helper classes for:

- Spacing (m-3, p-2)
- Alignment (text-center, justify-content-center)
- Colors (bg-primary, text-white)

JavaScript Interview Questions & Answers (1-100)

1. What is JavaScript?

Answer:
JavaScript is a **lightweight, interpreted, and dynamic** programming language used to create **interactive web pages**.

2. What are JavaScript data types?

Answer:
JavaScript has **7 primitive** types:

- String, Number, Boolean, Undefined, Null, Symbol, BigInt
- Plus **objects**, including Array, Function, and Date.

3. What is the difference between let, const, and var?

Answer:

Feature	**var**	**let**	**const**
Scope	Function	Block	Block
Re-declaration	Yes	No	No
Value Change	Yes	Yes	No

4. What is the difference between == and ===?

Answer:

- == → Checks **value only** (5 == "5" → true)
- === → Checks **value and type** (5 === "5" → false)

5. What is an arrow function in JavaScript?

Answer:
A shorthand way to write functions:

```
const add = (a, b) => a + b;
```

6. What are template literals in JavaScript?

Answer:
A way to embed variables inside strings using **backticks (`)**.

```
let name = "John";

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

7. What are JavaScript Promises?

Answer:
A way to handle asynchronous operations.

```
let promise = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Done!"), 1000);
});
```

8. What is async and await in JavaScript?

Answer:
A modern way to handle asynchronous code:

```
async function fetchData() {
  let data = await fetch('https://api.example.com');
}
```

9. What are JavaScript closures?

Answer:
A function that **remembers variables** from its outer scope:

```
function outer() {
  let count = 0;
  return function inner() {
    count++;
    console.log(count);
  };
}
let counter = outer();
counter(); // 1
counter(); // 2
```

10. What is the difference between null and undefined?

Answer:

- null → **Explicitly assigned** empty value
- undefined → A variable that **has not been assigned** a value

11. How do you declare an array in JavaScript?

Answer:

```
let arr = [1, 2, 3, 4, 5];
```

12. How do you loop through an array in JavaScript?

Answer:
Using forEach():

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

13. What is the difference between map() and forEach()?

Answer:

- map() → Returns a new array
- forEach() → Iterates without returning anything

14. What is localStorage in JavaScript?

Answer:
Stores data in the browser **permanently**:

localStorage.setItem("username", "John");

console.log(localStorage.getItem("username"));

15. What is the difference between sessionStorage and localStorage?

Answer:

- localStorage → Data **persists** even after closing the browser
- sessionStorage → Data **clears** when the session ends

16. What is this in JavaScript?

Answer:
Refers to the object that calls a function.

17. What is event bubbling and capturing in JavaScript?

Answer:

- **Bubbling** → Event starts from the **target** and propagates **upward**.
- **Capturing** → Event starts from the **top** and propagates **downward**.

18. What is setTimeout() in JavaScript?

Answer:
Executes code **after a delay**:

setTimeout(() => console.log("Hello"), 2000);

19. What is setInterval() in JavaScript?

Answer:
Repeats code **at intervals**:

setInterval(() => console.log("Repeating"), 1000);

20. What is the difference between apply(), call(), and bind()?

Answer:

Method	Usage
call()	func.call(obj, arg1, arg2)
apply()	func.apply(obj, [args])
bind()	let newFunc = func.bind(obj)

21. How do you remove duplicates from an array?

Answer:

```
let uniqueArr = [...new Set([1, 2, 2, 3, 4])];
```

22. What is the difference between synchronous and asynchronous JavaScript?

Answer:

- **Synchronous** → Code executes line by line
- **Asynchronous** → Tasks run **in the background** without blocking

23. What are JavaScript classes?

Answer:
A way to create objects:

```
class Person {
  constructor(name) {
    this.name = name;
  }
}
let p1 = new Person("John");
```

24. What is a callback function?

Answer:
A function passed as an argument:

```
function greet(callback) {
  callback();
}
greet(() => console.log("Hello"));
```

25. What is the difference between deep copy and shallow copy?

Answer:

- **Shallow Copy** → Copies reference
- **Deep Copy** → Creates a new object

26. What is the difference between Object.freeze() and Object.seal()?

Answer:

- Object.freeze() → Prevents **modification & addition**
- Object.seal() → Allows **modification but no addition**

27. How do you check if a variable is an array?

Answer:

Array.isArray([1, 2, 3]); // true

28. What is typeof in JavaScript?

Answer:
Returns the type of a variable:

console.log(typeof "Hello"); // string

29. What is the difference between function declaration and function expression?

Answer:

- **Declaration:**

function greet() {}

- **Expression:**

const greet = function() {};

30. What is the spread operator (...) in JavaScript?

Answer:
Used to expand an array:

let arr = [1, 2, 3];

let newArr = [...arr, 4, 5];

31. What is destructuring in JavaScript?

Answer:
A way to extract values from objects or arrays.

const person = { name: "John", age: 30 };

const { name, age } = person;

console.log(name, age);

32. What is the difference between splice() and slice()?

Answer:

Method	Modifies Original Array?	Usage
splice()	Yes	Removes/Replaces elements
slice()	No	Extracts part of an array

```
let arr = [1, 2, 3, 4];

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

arr.slice(1, 3); // [2, 3]
```

33. What is a higher-order function?

Answer:
A function that takes another function as an argument or returns a function.

```
function operate(func, x, y) {
  return func(x, y);
}

console.log(operate((a, b) => a + b, 5, 3)); // 8
```

34. What is the difference between push() and unshift()?

Answer:

- push() → Adds element **to the end**
- unshift() → Adds element **to the beginning**

```
let arr = [2, 3];

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

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

35. What is the difference between pop() and shift()?

Answer:

- pop() → Removes **last element**
- shift() → Removes **first element**

36. What is function hoisting in JavaScript?

Answer:
Functions **declared using function keyword** can be used before declaration.

```
sayHello(); // Works!

function sayHello() {
```

```
  console.log("Hello");
}
```

37. What is a pure function in JavaScript?

Answer:
A function that does **not change external variables** and returns the same output for the same input.

```
function add(a, b) {
  return a + b;
}
```

38. What is an immediately invoked function expression (IIFE)?

Answer:
A function that runs immediately after definition.

```
(function () {
  console.log("IIFE executed!");
})();
```

39. What is event delegation?

Answer:
A technique where a **parent element handles events** for child elements.

```
document.getElementById("parent").addEventListener("click", function (e) {
  if (e.target.tagName === "BUTTON") {
    console.log("Button clicked!");
  }
});
```

40. How do you prevent the default behavior of an event?

Answer:
Use .preventDefault().

```
document.querySelector("a").addEventListener("click", function (e) {
  e.preventDefault();
  console.log("Link click prevented!");
});
```

41. What is typeof null in JavaScript?

Answer:
It returns "object" (a known JavaScript bug).

42. What are getters and setters in JavaScript?

Answer:
Methods that allow controlled access to object properties.

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

43. What is the difference between .innerHTML and .textContent?

Answer:

- innerHTML → Parses HTML
- textContent → Sets/gets text without parsing HTML

44. What is the difference between querySelector() and getElementById()?

Answer:

- querySelector() → Returns **first match**
- getElementById() → Returns element **by ID**

45. What is a JavaScript generator function?

Answer:
A function that returns a sequence of values using yield.

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

46. What is debouncing in JavaScript?

Answer:
A technique to **limit function execution** in a short time span.

```
function debounce(func, delay) {
  let timer;
  return function () {
    clearTimeout(timer);
    timer = setTimeout(func, delay);
  };
}
```

47. What is throttling in JavaScript?

Answer:
A technique that ensures a function **runs at most once** in a given interval.

```
function throttle(func, limit) {
  let lastCall = 0;
  return function () {
    let now = new Date().getTime();
    if (now - lastCall >= limit) {
      func();
      lastCall = now;
    }
  };
}
```

48. How do you clone an object in JavaScript?

Answer:
Using Object.assign() or the spread operator:

```
let obj = { a: 1, b: 2 };
let clone1 = { ...obj };
let clone2 = Object.assign({}, obj);
```

49. How do you merge two objects in JavaScript?

Answer:
Using the spread operator or Object.assign().

```
let obj1 = { a: 1 };
```

```
let obj2 = { b: 2 };

let merged = { ...obj1, ...obj2 };
```

50. What is a WeakMap in JavaScript?

Answer:
A special kind of map where **keys must be objects** and are weakly referenced.

```
let weakMap = new WeakMap();

let obj = {};

weakMap.set(obj, "value");
```

51. What is a WeakSet in JavaScript?

Answer:
A collection of objects with **weak references**, meaning objects can be garbage collected.

```
let weakSet = new WeakSet();

let obj = { name: "John" };

weakSet.add(obj);

console.log(weakSet.has(obj)); // true
```

52. What is the difference between Set and WeakSet?

Answer:

Feature	**Set**	**WeakSet**
Stores	Any values	Only objects
Garbage Collection	No	Yes
Iteration	Yes	No

53. What is the difference between Map and WeakMap?

Answer:

- Map stores **any data types as keys**
- WeakMap stores **only objects as keys**

54. How do you convert a string to a number in JavaScript?

Answer:
Use parseInt(), parseFloat(), or the + operator.

```
console.log(parseInt("10")); // 10

console.log(parseFloat("10.5")); // 10.5

console.log(+"10"); // 10
```

55. How do you check if a variable is an array?

Answer:
Use Array.isArray().

```
console.log(Array.isArray([1, 2, 3])); // true
```

56. What is the difference between == and ===?

Answer:

- == checks **only value** (loose equality)
- === checks **value & type** (strict equality)

```
console.log(5 == "5");  // true

console.log(5 === "5"); // false
```

57. What are template literals in JavaScript?

Answer:
Strings enclosed in backticks (``) with ${} for interpolation.

```
let name = "John";

console.log(`Hello, ${name}!`); // Hello, John!
```

58. What is null vs undefined in JavaScript?

Answer:

Value	Meaning
null	Intentional absence of value
undefined	Variable declared but not assigned

59. What is a closure in JavaScript?

Answer:
A function that remembers variables from its outer scope even after execution.

```
function outer() {
  let count = 0;
  return function () {
    count++;
    console.log(count);
  };
}

let counter = outer();

counter(); // 1
```

```
counter(); // 2
```

60. What is the difference between apply(), call(), and bind()?

Answer:

- call() → Calls a function with **individual arguments**
- apply() → Calls a function with an **array of arguments**
- bind() → Returns a **new function**

```
function greet(greeting) {
  console.log(greeting + ", " + this.name);
}
let person = { name: "John" };
greet.call(person, "Hello"); // Hello, John
greet.apply(person, ["Hi"]); // Hi, John
let greetJohn = greet.bind(person, "Hey");
greetJohn(); // Hey, John
```

---

61. What is an arrow function?

Answer:
A shorter syntax for writing functions.

```
const add = (a, b) => a + b;
console.log(add(5, 3)); // 8
```

62. Why does this behave differently in arrow functions?

Answer:
Arrow functions **do not bind their own this**; they inherit it from the surrounding scope.

```
const obj = {
  name: "John",
  greet: () => console.log(this.name), // 'this' refers to the outer scope, not obj
};
obj.greet(); // undefined
```

63. What is the difference between localStorage, sessionStorage, and cookies?

Answer:

Storage Type	Data Expiry	Storage Limit	Accessible from
localStorage	Never	5MB	Same origin
sessionStorage	On tab close	5MB	Same tab
cookies	As set	4KB	Server & client

64. What is the purpose of the JSON.stringify() and JSON.parse() methods?

Answer:

- JSON.stringify() → Converts an object to a string
- JSON.parse() → Converts a string to an object

```
let obj = { name: "John" };

let str = JSON.stringify(obj);

console.log(JSON.parse(str)); // { name: "John" }
```

65. What is the event loop in JavaScript?

Answer:
A mechanism that handles **asynchronous operations** by running tasks in the call stack and waiting for events in the event queue.

66. What are Promises in JavaScript?

Answer:
An object that **handles asynchronous operations**.

```
let promise = new Promise((resolve, reject) => {

  setTimeout(() => resolve("Done!"), 1000);

});

promise.then(console.log); // Done!
```

67. What are async/await in JavaScript?

Answer:
A way to handle promises **synchronously**.

```
async function fetchData() {

  let data = await fetch("https://jsonplaceholder.typicode.com/todos/1");

  let result = await data.json();

  console.log(result);

}

fetchData();
```

68. What is a callback function?

Answer:
A function passed as an argument to another function.

```
function greet(name, callback) {
  console.log("Hello, " + name);
  callback();
}
greet("John", () => console.log("Callback executed!"));
```

69. What is the difference between synchronous and asynchronous JavaScript?

Answer:

- **Synchronous** → Executes line by line
- **Asynchronous** → Executes tasks in the background (e.g., setTimeout, fetch)

70. What is the purpose of the fetch() API?

Answer:
Used for making HTTP requests.

```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then(response => response.json())
  .then(data => console.log(data));
```

71. What is the typeof operator in JavaScript?

Answer:
Returns the data type of a variable.

```
console.log(typeof "Hello"); // string
console.log(typeof 42); // number
console.log(typeof true); // boolean
```

72. What is the instanceof operator?

Answer:
Checks if an object is an instance of a class.

```
console.log([] instanceof Array); // true
```

73. What are JavaScript modules?

Answer:
Reusable JavaScript files exported and imported using export and import.

```
// module.js
export function sayHello() {
  console.log("Hello!");
```

```
}


// main.js
import { sayHello } from "./module.js";
sayHello();
```

74. What is memoization in JavaScript?

Answer:
Caching function results for efficiency.

```
function memoize(fn) {
  let cache = {};
  return function (x) {
    if (cache[x]) return cache[x];
    cache[x] = fn(x);
    return cache[x];
  };
}
```

75. What is the difference between deep copy and shallow copy?

Answer:

- **Shallow Copy** → Copies only the first level
- **Deep Copy** → Copies nested objects

```
let obj1 = { a: { b: 1 } };
let obj2 = JSON.parse(JSON.stringify(obj1));
```

76. What is the difference between forEach(), map(), filter(), and reduce()?

Answer:

Method	Purpose	Returns
forEach()	Loops through elements	undefined
map()	Transforms each element	New array
filter()	Filters elements based on condition	New array
reduce()	Reduces array to single value	Single value

```
let numbers = [1, 2, 3, 4];
```

```
// forEach()
numbers.forEach(num => console.log(num * 2)); // 2, 4, 6, 8

// map()
let doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6, 8]

// filter()
let even = numbers.filter(num => num % 2 === 0);
console.log(even); // [2, 4]

// reduce()
let sum = numbers.reduce((acc, num) => acc + num, 0);
console.log(sum); // 10
```

---

77. What is destructuring in JavaScript?

Answer:
A way to extract values from arrays or objects.

```
// Array Destructuring
let [a, b] = [1, 2];
console.log(a, b); // 1 2

// Object Destructuring
let person = { name: "John", age: 30 };
let { name, age } = person;
console.log(name, age); // John 30
```

---

78. What are default parameters in JavaScript?

Answer:
Default values for function parameters.

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

```
}

greet(); // Hello, Guest!

greet("John"); // Hello, John!
```

---

79. How to deep clone an object in JavaScript?

Answer:
Use JSON.parse(JSON.stringify(obj)) or structured cloning.

```
let obj1 = { a: { b: 1 } };

let obj2 = JSON.parse(JSON.stringify(obj1));

console.log(obj2); // { a: { b: 1 } }
```

---

80. What is the difference between spread and rest operators?

Answer:

Operator	Purpose	Example
Spread ...	Expands elements	let arr2 = [...arr1]
Rest ...	Gathers elements	function sum(...args) {}

```
// Spread

let arr = [1, 2, 3];

let arr2 = [...arr, 4, 5];

console.log(arr2); // [1, 2, 3, 4, 5]


// Rest

function sum(...nums) {

  return nums.reduce((a, b) => a + b, 0);

}

console.log(sum(1, 2, 3, 4)); // 10
```

---

81. What is event delegation in JavaScript?

Answer:
A technique where a **single event listener** is used for multiple elements.

```
document.querySelector("#parent").addEventListener("click", function (event) {
```

```
  if (event.target.matches(".child")) {
    console.log("Child clicked!");
  }
});
```

---

82. How to prevent event bubbling?

Answer:
Use event.stopPropagation().

```
document.getElementById("child").addEventListener("click", function (event) {
  event.stopPropagation();
  console.log("Child clicked");
});
```

---

83. What is throttling and debouncing?

Answer:

- **Throttling** → Limits function execution at intervals.
- **Debouncing** → Delays execution until the user stops an action.

```
// Throttling
function throttle(func, limit) {
  let lastFunc, lastRan;
  return function () {
    if (!lastRan) {
      func.apply(this, arguments);
      lastRan = Date.now();
    } else {
      clearTimeout(lastFunc);
      lastFunc = setTimeout(() => {
        if (Date.now() - lastRan >= limit) {
          func.apply(this, arguments);
          lastRan = Date.now();
        }
```

```
    }, limit - (Date.now() - lastRan));
   }
 };
}

// Debouncing
function debounce(func, delay) {
 let timer;
 return function () {
  clearTimeout(timer);
  timer = setTimeout(() => func.apply(this, arguments), delay);
 };
}
```

---

84. What is the purpose of Object.freeze() and Object.seal()?

Answer:

Method	Prevents Adding	Prevents Deleting	Prevents Modifying
Object.freeze()	✅	✅	✅
Object.seal()	❌	✅	✅

```
let obj = { name: "John" };
Object.freeze(obj);
obj.name = "Doe"; // ❌ No effect
console.log(obj.name); // John
```

---

85. What is Intl API in JavaScript?

Answer:
Used for **internationalization** and **formatting** dates, numbers, etc.

```
let num = 1234567.89;
let formatted = new Intl.NumberFormat("en-US", { style: "currency", currency: "USD" }).format(num);
console.log(formatted); // $1,234,567.89
```

---

86. How does requestAnimationFrame() work?

Answer:
It **optimizes animations** by running before the next repaint.

```
function animate() {
  console.log("Frame rendered!");
  requestAnimationFrame(animate);
}
requestAnimationFrame(animate);
```

---

87. What is a generator function in JavaScript?

Answer:
A function that can be paused and resumed using yield.

```
function* generator() {
  yield 1;
  yield 2;
  yield 3;
}
let gen = generator();
console.log(gen.next().value); // 1
console.log(gen.next().value); // 2
```

---

88. How to remove duplicates from an array?

Answer:
Use Set.

```
let arr = [1, 2, 2, 3, 4, 4, 5];
let uniqueArr = [...new Set(arr)];
console.log(uniqueArr); // [1, 2, 3, 4, 5]
```

---

89. What is the difference between document.querySelector() and document.getElementById()?

Answer:

- querySelector() → Selects using **CSS selectors**
- getElementById() → Selects by **ID only**

```
document.querySelector("#myId");

document.getElementById("myId");
```

---

90. What is dynamic import in JavaScript?

Answer:
Imports modules **dynamically** instead of at the start.

```
import("./module.js").then(module => {

  module.sayHello();

});
```

91. What is Object.assign() in JavaScript?

Answer:
Object.assign() is used to **copy properties** from one or more objects to a target object.

```
let obj1 = { a: 1, b: 2 };

let obj2 = { c: 3 };


let mergedObj = Object.assign({}, obj1, obj2);

console.log(mergedObj); // { a: 1, b: 2, c: 3 }
```

---

92. What is Promise.all() in JavaScript?

Answer:
Promise.all() executes multiple promises **concurrently** and resolves when **all promises are resolved** or rejects if any fail.

```
let p1 = Promise.resolve(10);

let p2 = new Promise((resolve) => setTimeout(() => resolve(20), 1000));


Promise.all([p1, p2]).then(values => console.log(values)); // [10, 20]
```

---

93. What is Promise.race() in JavaScript?

Answer:
Promise.race() resolves or rejects as soon as **the first promise is settled**.

```
let p1 = new Promise(resolve => setTimeout(() => resolve("Fast"), 500));

let p2 = new Promise(resolve => setTimeout(() => resolve("Slow"), 1000));
```

```
Promise.race([p1, p2]).then(result => console.log(result)); // "Fast"
```

---

94. What is history.pushState() in JavaScript?

Answer:
history.pushState() is used to **modify the browser history** without refreshing the page.

```
history.pushState({ page: "home" }, "Home", "/home");

console.log(location.href); // Updates URL to '/home' without reloading
```

---

95. What is the difference between localStorage and sessionStorage?

Answer:

Feature	localStorage	sessionStorage
Data Lifetime	Permanent	Until tab closes
Scope	Across sessions	Per session only
Storage Limit	~5MB	~5MB

```
localStorage.setItem("name", "John");
console.log(localStorage.getItem("name")); // John


sessionStorage.setItem("age", "25");
console.log(sessionStorage.getItem("age")); // 25
```

---

96. What are Service Workers in JavaScript?

Answer:
Service workers run **in the background** to enable features like **offline caching and push notifications**.

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/sw.js").then(() => {
    console.log("Service Worker Registered");
  });
}
```

---

97. What is a Symbol in JavaScript?

Answer:
A Symbol is a **unique and immutable identifier** used to avoid property name conflicts.

```
let sym = Symbol("id");

let obj = { [sym]: 123 };


console.log(obj[sym]); // 123

console.log(Symbol("id") === Symbol("id")); // false
```

---

98. What is BigInt in JavaScript?

Answer:
BigInt allows handling **very large integers** beyond Number.MAX_SAFE_INTEGER.

```
let bigNumber = 123456789012345678901234567890n;

console.log(bigNumber + 10n); // 123456789012345678901234567900n
```

---

99. What is navigator.geolocation in JavaScript?

Answer:
It is used to **get the user's location** with their permission.

```
navigator.geolocation.getCurrentPosition(position => {
  console.log(position.coords.latitude, position.coords.longitude);
});
```

---

100. What is window.requestIdleCallback() in JavaScript?

Answer:
It **executes a function when the browser is idle**, improving performance.

```
window.requestIdleCallback(() => {
  console.log("Executed when browser is idle");
});
```