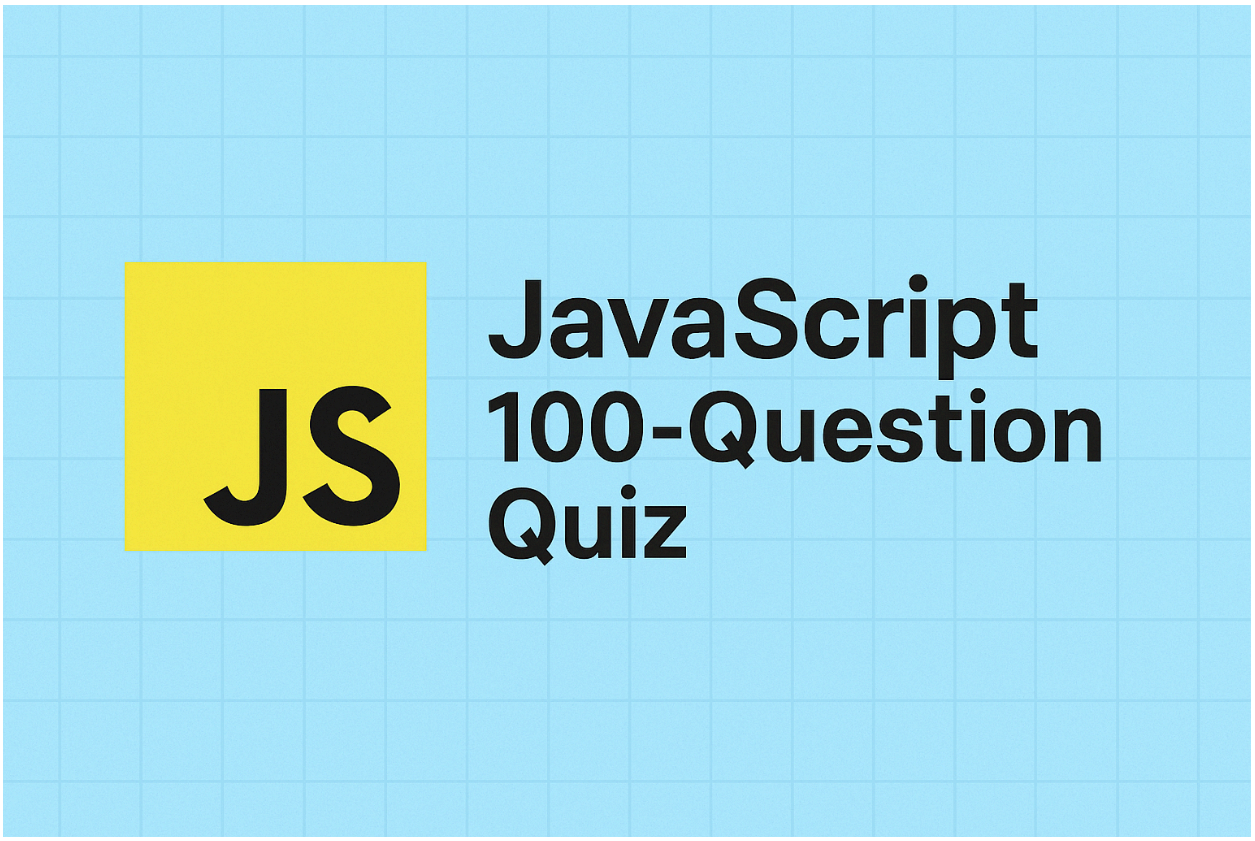

1. What does `typeof 42` return?

A. `"int"`
B. `"number"`
C. `"integer"`
D. `"numeric"`

Correct answer: B
Explanation: In JavaScript, all numeric values (integers and floats) share the type `"number"`.

---

2. What is the result of `typeof null`?

A. `"null"`
B. `"undefined"`
C. `"object"`
D. `"value"`

Correct answer: C
Explanation: This is a long-standing JavaScript quirk: `typeof null` returns `"object"` even though `null` is a primitive.

3. What will `console.log(1 + "2")` output?

A. `3`
B. `"3"`
C. `"12"`
D. `NaN`

Correct answer: C
Explanation: When adding a number and a string, JavaScript converts the number to a string and concatenates, resulting in `"12"`.

---

4. What does `NaN` stand for?

A. Not a Null
B. Not a Number
C. Negative a Number
D. No actual Number

Correct answer: B
Explanation: `NaN` stands for “Not a Number” and represents an invalid numeric result (e.g., `0 / 0`).

---

5. Which of these is not a primitive type in JavaScript?

A. `string`
B. `boolean`
C. `object`
D. `symbol`

Correct answer: C
Explanation: `object` is not primitive. Primitive types are `string`, `number`, `boolean`, `null`, `undefined`, `symbol`, and `bigint`.

---

6. What will `typeof undefined` return?

A. `"null"`
B. `"undefined"`
C. `"object"`
D. `"void"`

Correct answer: B
Explanation: The primitive value `undefined` has the type `"undefined"`.

---

7. Which operator is used for strict equality comparison?

A. `==`
B. `=`
C. `===`
D. `=>`

Correct answer: C
Explanation: `===` compares both value and type, with no type coercion. `==` allows coercion.

---

8. What is the result of `"5" == 5`?

A. `true`
B. `false`
C. `NaN`
D. Throws an error

Correct answer: A
Explanation: `==` coerces types, so the string `"5"` is converted to number `5`, making the comparison `true`.

---

9. What is the result of `"5" === 5`?

A. `true`
B. `false`
C. `NaN`
D. Throws an error

Correct answer: B
Explanation: Strict equality (`===`) compares type and value, and `"5"` (string) is not the same type as `5` (number).

---

10. Which keyword declares a block-scoped variable?

A. `var`
B. `let`

C. `const`
D. Both B and C

Correct answer: D
Explanation: Both `let` and `const` are block-scoped; `var` is function-scoped.

---

11. What happens if you access a variable declared with `let` before its declaration?

A. Returns `undefined`
B. Throws a ReferenceError
C. Returns `null`
D. Returns an empty string

Correct answer: B
Explanation: `let` and `const` are in the temporal dead zone before declaration and cause a `ReferenceError` when accessed.

---

12. What will this code log?

```
console.log(a);
var a = 10;
```

A. `10`
B. `undefined`
C. `null`
D. ReferenceError

Correct answer: B
Explanation: `var` is hoisted with an initial value of `undefined`, so the log shows `undefined`.

---

13. What will this code log?

```
console.log(b);
let b = 10;
```

A. `10`
B. `undefined`
C. `null`
D. ReferenceError

Correct answer: D
Explanation: `let` is hoisted but not initialized, causing a `ReferenceError` if accessed before the declaration.

---

14. What is a closure?

A. A way to close opened files
B. A function bundled with its lexical environment
C. A function that always returns another function
D. A method to end a program

Correct answer: B
Explanation: A closure is created when an inner function retains access to variables from its outer function's scope, even after the outer function has returned.

---

15. Which array method adds one or more elements to the end of an array?

A. `push()`
B. `pop()`
C. `shift()`
D. `unshift()`

Correct answer: A
Explanation: `push()` appends elements to the end; `pop()` removes from end, `shift()` from start, `unshift()` adds to start.

---

16. Which array method creates a new array without modifying the original?

A. `push()`
B. `splice()`
C. `slice()`
D. `pop()`

Correct answer: C
Explanation: `slice()` returns a new array containing a portion of the original; `splice()` mutates the original.

---

17. What is the result of `[1, 2, 3].length`?

A. `2`
B. `3`
C. `4`
D. `undefined`

Correct answer: B
Explanation: The `length` property returns the number of elements in the array, which is 3.

---

18. Which method converts a JSON string into an object?

A. `JSON.encode()`
B. `JSON.parse()`
C. `JSON.stringify()`
D. `JSON.toObject()`

Correct answer: B
Explanation: `JSON.parse()` parses a JSON string into a JavaScript object; `JSON.stringify()` does the opposite.

---

19. What is the output?

```
console.log(0 == false);
```

A. `true`
B. `false`
C. `NaN`
D. Throws error

Correct answer: A
Explanation: `==` performs type coercion; `false` is coerced to `0`, so the comparison is `true`.

---

20. What is the output?

```
console.log(0 === false);
```

A. `true`
B. `false`
C. `NaN`
D. Throws error

Correct answer: B
Explanation: Strict equality compares type and value; `0` (number) and `false` (boolean) are different types.

---

21. How do you write an arrow function that returns the square of `x`?

A. `x => { x * x }`
B. `x => x * x`
C. `(x) => return x * x`
D. `x -> x * x`

Correct answer: B
Explanation: For single-expression arrow functions, you can omit braces and `return`; the expression result is returned.

---

22. What will this output?

```
console.log(typeof (() => {}));
```

A. `"function"`
B. `"object"`
C. `"arrow"`
D. `"undefined"`

Correct answer: A
Explanation: Arrow functions are still functions in JavaScript, so `typeof` returns `"function"`.

---

23. Which of these is not a valid way to define a function?

A. Function declaration
B. Function expression
C. Arrow function
D. Class function literal

Correct answer: D
Explanation: "Class function literal" is not a standard term. Functions can be defined via declarations, expressions, or arrow syntax.

---

24. What does `Array.isArray(value)` do?

A. Checks if value is iterable
B. Checks if value is an array
C. Checks if value is an object
D. Converts value to array

Correct answer: B
Explanation: `Array.isArray()` returns `true` only when the value is an actual array.

---

25. What is the value of `Number("hello")`?

A. `"hello"`
B. `NaN`
C. `0`
D. `undefined`

Correct answer: B
Explanation: Converting a non-numeric string to number results in `NaN`.

---

26. Which of these is falsy?

A. `"0"`
B. `[]`
C. `{}`
D. `""`

Correct answer: D
Explanation: The empty string `""` is falsy; `"0"`, `[]`, and `{}` are truthy.

---

27. Which keyword stops a loop immediately?

A. `exit`
B. `stop`
C. `break`
D. `return`

Correct answer: C
Explanation: `break` exits the nearest loop or switch statement.

---

28. Which keyword skips the current iteration of a loop and continues with the next?

A. `skip`
B. `next`
C. `continue`
D. `pass`

Correct answer: C
Explanation: `continue` stops the current iteration and moves to the next one in the loop.

---

29. What is the result?

```
console.log("5" - 2);
```

A. `"52"`
B. `3`
C. `NaN`
D. `"3"`

Correct answer: B
Explanation: The `-` operator triggers numeric coercion; `"5"` becomes number `5`, so `5 - 2` is `3`.

---

30. What is the result?

```
console.log("5" + 2);
```

A. `"52"`
B. `7`
C. `NaN`
D. `"7"`

Correct answer: A
Explanation: `+` acts as string concatenation when one operand is a string, resulting in `"52"`.

---

31. What is `window` in a browser?

A. A built-in array
B. The global object for browser JavaScript

C. A reserved variable name for DOM nodes
D. A CSS object

Correct answer: B
Explanation: In browsers, the global scope is represented by the `window` object, which holds global variables and APIs.

---

32. In strict mode (`"use strict"`), what happens if you assign to an undeclared variable?

A. It creates a global variable
B. It creates a local variable
C. It throws a ReferenceError
D. It is silently ignored

Correct answer: C
Explanation: Strict mode prevents implicit global variable creation and throws a `ReferenceError`.

---

33. How do you enable strict mode in a script file?

A. `enable strict;`
B. `"use strict";` at the top
C. `use strict;` without quotes
D. `strict_mode(true);`

Correct answer: B
Explanation: The directive `"use strict";` at the top of a script or function enables strict mode.

---

34. What is the output?

```
console.log([] == false);
```

A. `true`
B. `false`
C. `NaN`
D. Throws error

Correct answer: A
Explanation: With `==`, `[]` is coerced to `""`, then to `0`; `false` is also coerced to `0`, so comparison is `true`. It's a classic weirdness.

35. What is the value of `typeof NaN`?

A. `"nan"`
B. `"number"`
C. `"undefined"`
D. `"object"`

Correct answer: B
Explanation: `NaN` is a special numeric value, so its type is `"number"`.

---

36. How do you check if a value is `NaN` (and only `NaN`) in modern JavaScript?

A. `value == NaN`
B. `value === NaN`
C. `isNaN(value)`
D. `Number.isNaN(value)`

Correct answer: D
Explanation: `Number.isNaN()` checks specifically for `NaN` without coercion; `isNaN()` coerces and can give surprising results.

---

37. What is the result?

```
console.log([] + []);
```

A. `[]`
B. `""`
C. `"[]"`
D. `NaN`

Correct answer: B
Explanation: Arrays are converted to strings when using `+`. `[].toString()` is `""`, so empty string + empty string is `""`.

---

38. What does `Array.prototype.map()` return?

A. A new array
B. A modified original array

C. A number
D. An object with keys and values

Correct answer: A
Explanation: `map()` creates a new array containing the results of applying a callback to each element, without mutating the original.

---

39. Which method is best for filtering elements from an array based on a condition?

A. `forEach()`
B. `map()`
C. `filter()`
D. `reduce()`

Correct answer: C
Explanation: `filter()` returns a new array containing only the elements for which the callback returns `true`.

---

40. What does `Array.prototype.forEach()` return?

A. A new array
B. The original array
C. `undefined`
D. The number of iterations

Correct answer: C
Explanation: `forEach()` is purely for side effects; it always returns `undefined`.

---

41. What does `Array.prototype.reduce()` typically do?

A. Sorts an array
B. Flattens nested arrays only
C. Reduces an array to a single value
D. Filters elements

Correct answer: C
Explanation: `reduce()` accumulates results by applying a callback, producing a single output value (sum, object, etc.).

---

42. Which operator spreads the elements of an iterable?

A. `...` (spread operator)
B. `*`
C. `&`
D. `=>`

Correct answer: A
Explanation: The spread syntax `...iterable` expands its elements in array literals, function calls, etc.

---

43. What is destructuring in JavaScript?

A. Removing properties from objects
B. Assigning properties or elements from objects/arrays into variables
C. Deleting variables
D. Breaking a loop

Correct answer: B
Explanation: Destructuring lets you extract values from arrays or objects into distinct variables using pattern syntax.

---

44. Which is valid array destructuring?

A. `let {a, b} = [1, 2];`
B. `let [a, b] = [1, 2];`
C. `let (a, b) = [1, 2];`
D. `let [a: 1, b: 2];`

Correct answer: B
Explanation: Array destructuring uses square brackets, so `[a, b]` matches elements of the array.

---

45. Which is valid object destructuring?

A. `let [name] = { name: "Max" };`
B. `let {name} = { name: "Max" };`
C. `let (name) = { name: "Max" };`
D. `let {name: "Max"};`

Correct answer: B
Explanation: Object destructuring uses curly braces with property names: `{name} = obj`.

---

46. What is the default value of `this` inside a regular function (non–strict mode) called as `fn()` in the browser?

A. `undefined`
B. The global object (`window`)
C. The function object itself
D. `null`

Correct answer: B
Explanation: In non-strict mode, a plain function call binds `this` to the global object (`window` in browsers).

---

47. In an arrow function, `this` is:

A. Dynamically bound
B. Always `window`
C. Lexically inherited from the surrounding scope
D. Always `undefined`

Correct answer: C
Explanation: Arrow functions don't have their own `this`; they capture `this` from the enclosing lexical scope.

---

48. How do you create a new object using a constructor function?

A. `Person()`
B. `new Person()`
C. `create Person()`
D. `Object(Person)`

Correct answer: B
Explanation: The `new` keyword creates a new object and binds `this` inside the constructor function to that object.

---

49. Which prototype is used when calling a method on an array literal like `[]`?

A. `Object.prototype`
B. `Array.prototype`

C. `Function.prototype`
D. `Prototype.prototype`

Correct answer: B
Explanation: Arrays inherit methods (like `push`, `map`) from `Array.prototype`, which itself inherits from `Object.prototype`.

---

50. What does `Object.create(proto)` do?

A. Copies all properties from `proto`
B. Creates an object with its internal prototype set to `proto`
C. Clones `proto` deeply
D. Creates a new class

Correct answer: B
Explanation: `Object.create(proto)` returns a new object whose `[[Prototype]]` is `proto`.

---

51. How do you define a class in modern JavaScript?

A. `class Person {}`
B. `function class Person {}`
C. `Person class {}`
D. `new class Person {}`

Correct answer: A
Explanation: ES6 introduced the `class` syntax: `class Name { ... }`.

---

52. How do you define a method inside a class?

A. `methodName: function() {}`
B. `function methodName() {}`
C. `methodName() {}`
D. `let methodName() {}`

Correct answer: C
Explanation: Inside class bodies, methods are defined as `methodName() { ... }` without `function` keyword.

---

53. How do you create a subclass from a class `Parent`?

A. `class Child: Parent {}`
B. `class Child extends Parent {}`
C. `class Child inherits Parent {}`
D. `class Child Parent {}`

Correct answer: B
Explanation: `extends` is used to define a derived class: `class Child extends Parent`.

---

54. Which keyword calls the parent class constructor?

A. `this()`
B. `parent()`
C. `base()`
D. `super()`

Correct answer: D
Explanation: `super()` calls the constructor of the parent class in a subclass.

---

55. What is the output?

```
console.log(typeof function() {});
```

A. `"object"`
B. `"function"`
C. `"callable"`
D. `"method"`

Correct answer: B
Explanation: Regular functions have the type `"function"` when using `typeof`.

---

56. Which statement about promises is true?

A. A promise can be pending, fulfilled, or rejected
B. A promise can only be fulfilled
C. Promises block the main thread
D. Promises replace all callbacks

Correct answer: A
Explanation: Promises have three main states: `pending`, `fulfilled`, and `rejected`. They're a pattern for handling async operations.

---

57. What does `Promise.resolve(5)` create?

A. A rejected promise with value 5
B. A fulfilled promise with value 5
C. A pending promise with value 5
D. A synchronous return value 5

Correct answer: B
Explanation: `Promise.resolve(5)` returns a promise that is immediately fulfilled with the value `5`.

---

58. How do you attach a success handler to a promise `p`?

A. `p.then(onFulfilled)`
B. `p.success(onFulfilled)`
C. `p.done(onFulfilled)`
D. `p.resolve(onFulfilled)`

Correct answer: A
Explanation: `then()` registers callbacks for fulfilled and/or rejected states.

---

59. How do you handle errors in a promise chain?

A. With `try/catch` only
B. With `.catch()`
C. With `.error()`
D. With `.fail()`

Correct answer: B
Explanation: `.catch()` handles rejected promises and errors thrown in previous `.then()` handlers.

---

60. Which keyword is used to make a function asynchronous?

A. `sync`
B. `await`

C. `async`
D. `defer`

Correct answer: C
Explanation: Prefixing a function with `async` makes it return a promise and allows `await` inside it.

---

61. What does `await` do in an `async` function?

A. Pauses the whole program
B. Pauses only that async function until the promise settles
C. Converts a promise to a callback
D. Makes a function synchronous

Correct answer: B
Explanation: `await` suspends the async function execution until the promise resolves or rejects, without blocking the main thread.

---

62. What happens if an error is thrown inside an `async` function?

A. It crashes the browser
B. It becomes a rejected promise
C. It is ignored
D. It becomes a fulfilled promise

Correct answer: B
Explanation: Errors thrown inside `async` functions reject the returned promise, which can be caught with `.catch()` or `try/catch` around `await`.

---

63. What does `setTimeout(fn, 0)` do?

A. Executes `fn` immediately
B. Schedules `fn` after the current call stack clears
C. Blocks execution until `fn` completes
D. Throws an error

Correct answer: B
Explanation: `setTimeout(fn, 0)` queues `fn` to run after the current call stack and microtasks, not truly instantly.

---

64. Which of these is not part of the JavaScript language itself but provided by the browser?

A. `Array`
B. `Promise`
C. `document`
D. `Object`

Correct answer: C
Explanation: `document` is part of the DOM API provided by the browser, not the core language.

---

65. How do you select an element with id `"main"` in the DOM?

A. `document.getElement("main")`
B. `document.getElementById("main")`
C. `document.query("#main")`
D. `document.id("main")`

Correct answer: B
Explanation: `getElementById()` selects an element whose `id` matches the given string.

---

66. Which method selects the first element matching a CSS selector?

A. `document.querySelector()`
B. `document.querySelectorAll()[0]` only
C. `document.getElementBySelector()`
D. `document.selectFirst()`

Correct answer: A
Explanation: `querySelector()` returns the first matching element or `null`.

---

67. How do you add a click event listener to a button element `btn`?

A. `btn.on("click", fn)`
B. `btn.click(fn)`
C. `btn.addEventListener("click", fn)`
D. `btn.addClick(fn)`

Correct answer: C
Explanation: `addEventListener()` is the standard way to listen to events on DOM elements.

---

68. What will `document.querySelectorAll(".item")` return?

A. A single element
B. An array
C. A NodeList
D. A string

Correct answer: C
Explanation: `querySelectorAll()` returns a `NodeList` (array-like collection) of all matching elements.

---

69. Which property changes the text inside an element?

A. `innerText`
B. `text`
C. `content`
D. `valueText`

Correct answer: A
Explanation: `innerText` (or `textContent`) changes the textual content of an element.

---

70. How do you prevent a form's default submit behavior?

A. `event.stop()`
B. `event.preventDefault()`
C. `event.cancel()`
D. `event.stopPropagation()`

Correct answer: B
Explanation: `preventDefault()` stops the default action (like form submission or link navigation).

---

71. What does `event.stopPropagation()` do?

A. Prevents default browser behavior
B. Stops the event from bubbling up to parent elements

C. Disables all event listeners
D. Cancels form submission

Correct answer: B
Explanation: `stopPropagation()` prevents the event from moving to ancestor elements in the DOM event flow.

---

72. What is hoisting?

A. Moving files to the top of a project
B. JavaScript's behavior of moving declarations to the top of their scope
C. Sorting variables alphabetically
D. Loading external scripts

Correct answer: B
Explanation: Declarations (with `var`, function declarations) are hoisted to the top of their scope before execution.

---

73. Which are hoisted with their definitions?

A. Function declarations
B. Function expressions
C. Arrow functions assigned to variables
D. Both B and C

Correct answer: A
Explanation: Function declarations are hoisted with their full definitions. Function expressions and arrow functions are hoisted only as variables.

---

74. What is the output?

```
console.log(hoisted());
function hoisted() {
  return "Hello";
}
```

A. `"Hello"`
B. `undefined`
C. Error
D. `null`

Correct answer: A
Explanation: The function declaration `hoisted` is hoisted entirely, so it can be called before it appears in the code.

---

75. What is the output?

```
console.log(x);
let x = 5;
```

A. `5`
B. `undefined`
C. `null`
D. ReferenceError

Correct answer: D
Explanation: `let` variables cannot be accessed before declaration due to the temporal dead zone.

---

76. Which method checks if an array includes a certain value?

A. `arr.contains(value)`
B. `arr.has(value)`
C. `arr.includes(value)`
D. `arr.exists(value)`

Correct answer: C
Explanation: `includes()` returns `true` if the array contains the value.

---

77. Which statement about `const` is true?

A. `const` variables cannot be reassigned or mutated
B. `const` prevents reassigning the variable binding but objects can still be mutated
C. `const` makes values deeply immutable
D. `const` is function-scoped

Correct answer: B
Explanation: `const` prevents rebinding the variable, but if it references an object/array, its internal properties can still change.

---

78. What is the output?

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

A. `1`
B. `2`
C. Error
D. `undefined`

Correct answer: B
Explanation: The reference `obj` is constant, but its contents are mutable, so `a` can change from 1 to 2.

---

79. What is a template literal?

A. A precompiled string
B. A string defined with backticks that can include expressions with `${}`
C. A JSON template
D. A string used only for HTML

Correct answer: B
Explanation: Template literals use backticks (`` ` ``) and allow interpolation with `${expression}` and multi-line strings.

---

80. Which is a valid template literal?

A. `"Hello ${name}"`
B. `'Hello ${name}'`
C. `` `Hello ${name}` ``
D. `Hello ${name}`

Correct answer: C
Explanation: Interpolation with `${}` only works inside backtick-delimited template literals.

---

81. How do you export a named function in ES modules?

A. `module.export function myFunc(){}`
B. `export function myFunc() {}`

C. `exports.myFunc = function(){}` in browser modules
D. `export: myFunc()`

Correct answer: B
Explanation: ES module syntax uses `export` before declarations, e.g., `export function myFunc() {}`.

---

82. How do you import a named function `myFunc` from `./utils.js`?

A. `import { myFunc } from "./utils.js";`
B. `require("./utils.js").myFunc;` in browser modules
C. `import myFunc from "./utils.js";` only
D. `include myFunc from "./utils.js";`

Correct answer: A
Explanation: Named imports use curly braces: `import { myFunc } from "./utils.js";`.

---

83. What is the default export import syntax?

A. `import { default } from "./mod.js";`
B. `import * as default from "./mod.js";`
C. `import something from "./mod.js";`
D. `import default("./mod.js");`

Correct answer: C
Explanation: `import something from "./mod.js";` imports the module's default export as `something`.

---

84. Which tool is used at runtime to determine if a property exists directly on an object (not in its prototype chain)?

A. `in` operator
B. `obj.hasOwnProperty("prop")`
C. `obj.propertyExists("prop")`
D. `Object.exists(obj, "prop")`

Correct answer: B
Explanation: `hasOwnProperty` checks only own (non-inherited) properties; `in` checks the entire prototype chain.

85. What is the result?

```
const a = { x: 1 };
const b = a;
b.x = 2;
console.log(a.x);
```

A. `1`
B. `2`
C. `undefined`
D. Error

Correct answer: B
Explanation: `a` and `b` reference the same object, so changing `b.x` also changes `a.x`.

---

86. How do you make a shallow copy of an object `obj`?

A. `const copy = obj;`
B. `const copy = Object.copy(obj);`
C. `const copy = { ...obj };`
D. `const copy = new obj();`

Correct answer: C
Explanation: The spread operator `{ ...obj }` creates a shallow copy of the object's own enumerable properties.

---

87. What is event delegation?

A. Assigning one event to multiple elements at once
B. Attaching a single event listener to a parent element to handle events from its children
C. Delegating events from browser to server
D. Combining multiple events into one

Correct answer: B
Explanation: Event delegation uses event bubbling so one listener on a parent can handle events from many child elements efficiently.

---

88. Which built-in data structure maintains insertion order and uses key-value pairs with any type of key?

A. `Object`
B. `Map`
C. `Set`
D. `Array`

Correct answer: B
Explanation: `Map` allows keys of any type and preserves insertion order; `Object` keys are strings/symbols.

---

89. Which structure holds unique values only (no duplicates)?

A. `Array`
B. `Map`
C. `Set`
D. `Object`

Correct answer: C
Explanation: `Set` stores unique values; adding the same value again has no effect.

---

90. What does `"use strict"` mainly help with?

A. Faster network requests
B. Cleaner syntax highlighting
C. Catching common mistakes and unsafe actions
D. Making code run in parallel

Correct answer: C
Explanation: Strict mode throws errors for unsafe actions (like implicit globals) and disallows some problematic features.

---

91. What does `Symbol()` create?

A. A unique, immutable value usable as an object key
B. A string alias
C. A new data type like number
D. A private variable

Correct answer: A
Explanation: Symbols are unique, immutable primitive values often used as object property keys.

---

92. What is the difference between `==` and `===`?

A. `===` compares only types
B. `==` compares only values
C. `===` compares value and type without coercion
D. They are identical

Correct answer: C
Explanation: `===` is strict equality (no coercion); `==` allows type coercion.

---

93. What does `Object.freeze(obj)` do?

A. Prevents adding, removing, or changing properties
B. Prevents only adding new properties
C. Prevents only deleting properties
D. Makes deep immutable copies

Correct answer: A
Explanation: A frozen object's existing properties cannot be changed, added, or deleted (shallow freeze).

---

94. What is the output?

```
console.log(typeof [].constructor);
```

A. `"array"`
B. `"object"`
C. `"function"`
D. `"constructor"`

Correct answer: C
Explanation: `[].constructor` is `Array`, which is a function, so `typeof` is `"function"`.

---

95. What does `Object.keys(obj)` return?

A. All values of `obj`
B. An array of `obj`'s own enumerable property names
C. Prototype chain keys
D. A Map of key/value entries

Correct answer: B
Explanation: `Object.keys()` returns an array with the object's own enumerable property names.

---

96. What is the main difference between `for...in` and `for...of`?

A. `for...in` iterates values; `for...of` keys
B. `for...in` iterates keys; `for...of` values of iterables
C. They are the same
D. `for...of` only works on objects

Correct answer: B
Explanation: `for...in` loops over enumerable property names (keys), while `for...of` iterates over iterable values (arrays, strings, etc.).

---

97. What will this log?

```
console.log("2" * "3");
```

A. `"23"`
B. `6`
C. `NaN`
D. `"6"`

Correct answer: B
Explanation: The `*` operator coerces both strings to numbers, so `"2"` and `"3"` become `2` and `3`, resulting in `6`.

---

98. What is the output?

```
console.log(Boolean("false"));
```

A. `true`
B. `false`
C. `NaN`
D. Error

Correct answer: A
Explanation: Non-empty strings are truthy, so `"false"` converts to `true` as a boolean.

---

99. What is function currying?

A. Combining two functions into one
B. Transforming a function with multiple arguments into a series of functions each taking one argument
C. Removing parameters from a function
D. Overriding built-in functions

Correct answer: B
Explanation: Currying breaks down a multi-argument function into nested unary functions, enabling partial application.

---

100. What is the main purpose of JavaScript in web development?

A. Styling web pages
B. Structuring content
C. Adding interactivity and dynamic behavior
D. Serving web pages from the server

Correct answer: C
Explanation: HTML structures content, CSS styles it, and JavaScript adds logic, interactivity, and dynamic updates on the client side.