1. Q:What is JavaScript?
   A: JavaScript is a lightweight, interpreted scripting language used to make web pages interactive.
2. Q:Who developed JavaScript?
   A: Brendan Eich developed it at Netscape in 1995.
3. Q: Is JavaScript the same as Java?
   A: No, they are different languages with different use cases.
4. Q:Where is JavaScript used?
   A: It's used in web development, servers (Node.js), games, and apps.
5. Q: How do you include JavaScript in an HTML file?
   A: Using the <script> tag.
6. **Q:** What is the default scripting language in browsers?
   A: JavaScript.
7. **Q:** What is the file extension of JavaScript?
   A: .js
8. **Q:** Can JavaScript run outside the browser?
   A: Yes, using environments like Node.js.
9. **Q:** What are the two main types of comments in JavaScript?
   A: Single-line (//) and multi-line (/* */).
10. **Q:** What is the output of console.log("Hello" + " World")?
    A: "Hello World"
11. **Q:** What is a variable?
    A: A container for storing data values.
12. **Q:** What are the keywords to declare variables?
    A: var, let, and const.
13. **Q:** What is the difference between let and const?
    A: let can be reassigned; const cannot.

14. **Q:** Is JavaScript case-sensitive?
  A: Yes.

15. **Q:** What is a keyword?
  A: A reserved word that has a special meaning in JavaScript.

16. **Q:** How do you show an alert in JavaScript?
  A: Using alert("message").

17. **Q:** How do you write to the console?
  A: console.log("message")

18. **Q:** What is a data type?
  A: It defines the type of value a variable holds.

19. **Q:** Name some primitive data types in JavaScript.
  A: String, Number, Boolean, Null, Undefined, Symbol, BigInt.

20. **Q:** What is typeof operator used for?
  A: To check the data type of a variable.

21. **Q:** What is the output of typeof null?
  A: "object" (this is a known quirk in JS)

22. **Q:** What is the output of typeof NaN?
  A: "number"

23. **Q:** What is hoisting in JavaScript?
  A: Variables and functions are moved to the top of their scope during compilation.

24. **Q:** Can you declare a variable without using var, let, or const?
  A: Yes, but it becomes a global variable (not recommended).

25. **Q:** What is the difference between == and ===?
  A: == compares values, === compares value and type.

26. **Q:** What is the output of 10 + "5"?
  A: "105" (number + string = string)

26. **Q:** What is the output of "5" - 2?
   A: 3 (JavaScript converts string to number)
27. **Q:** What does NaN stand for?
   A: Not a Number.
28. **Q:** What is the output of typeof NaN?
   A: "number"
29. **Q:** How do you check if a value is NaN?
   A: Using isNaN(value)
30. **Q:** What is a Boolean in JavaScript?
   A: A logical data type with two values: true and false.
31. **Q:** What is the output of Boolean(0)?
   A: false
32. **Q:** What is the output of Boolean("0")?
   A: true (non-empty string is truthy)
33. **Q:** What values are considered falsy in JavaScript?
   A: false, 0, "", null, undefined, NaN
34. **Q:** How do you write a single-line comment?
   A: // comment
35. **Q:** How do you write a multi-line comment?
   A: /* comment */
36. **Q:** Can a JavaScript variable name start with a number?
   A: No.
37. **Q:** Can JavaScript variable names contain $ or _?
   A: Yes.
38. **Q:** What is the difference between null and undefined?
   A: null is an assigned value; undefined means a variable has been declared but not assigned.

39. **Q:** What does parseInt("10px") return?
A: 10

40. **Q:** What does parseFloat("10.5kg") return?
A: 10.5

41. **Q:** What does isNaN("hello") return?
A: true

42. **Q:** What is the output of typeof undefined?
A: "undefined"

43. **Q:** What is the output of typeof function() {}?
A: "function"

44. **Q:** What is the output of typeof []?
A: "object"

45. **Q:** What is the output of typeof {}?
A: "object"

46. **Q:** How do you create an array in JavaScript?
A: Using square brackets: let arr = [1, 2, 3];

47. **Q:** How do you create an object in JavaScript?
A: Using curly braces: let obj = {name: "John", age: 30};

48. **Q:** How do you check the length of a string?
A: Using .length, e.g., "hello".length

49. **Q:** How do you convert a string to a number?
A: Using Number("123") or parseInt("123")

50. **Q:** How do you convert a number to a string?
A: Using .toString() or String(123)

51. **Q:** What is a template literal?
A: A string with embedded expressions using backticks (` Hello ${name}`)

52. **Q:** How do you create a function in JavaScript?
A:

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

54. **Q:** What is an anonymous function?
A: A function without a name.

55. **Q:** What is a callback function?
A: A function passed as an argument to another function.

56. **Q:** What is the use of return in a function?
A: It returns a value from the function.

57. **Q:** Can a function return multiple values?
A: Not directly, but it can return an array or object.

58. **Q:** What are function parameters?
A: Inputs declared in the function definition.

59. **Q:** What are function arguments?
A: Values passed to the function when it is called.

60. **Q:** What is the default return value of a function?
A: undefined (if no return is specified)

61. **Q:** Can you assign a function to a variable?
A: Yes. Functions are first-class citizens.

62. **Q:** What is an arrow function?
A: A shorter syntax for writing functions:

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

63. **Q:** What is the purpose of the console.log() function?
A: To print messages to the browser console.

64. **Q:** What is the use of the alert() method?
   A: To display a popup message to the user.

65. **Q:** What is the output of "5" == 5?
   A: true (type coercion)

66. **Q:** What is the output of "5" === 5?
   A: false (strict comparison)

67. **Q:** What is type coercion?
   A: Automatic conversion of data types.

68. **Q:** What is a ternary operator?
   A: A shorthand for if-else:

condition ? expr1 : expr2;

69. **Q:** What is short-circuit evaluation?
   A: Logical operations stop once the result is known:

true || anything → true

false && anything → false

70. **Q:** What is the use of break in loops?
   A: To exit the loop early.

71. **Q:** What is the use of continue in loops?
   A: To skip the current iteration.

72. **Q:** What is the difference between for and while loop?
   A: for is used when the number of iterations is known; while is used when the condition is evaluated each time.

73. **Q:** How do you stop an infinite loop?
   A: Add a proper exit condition or use break.

74. **Q:** What is the output of !!"text"?
   A: true (double negation converts to Boolean)

75. **Q:** What is an expression in JavaScript?
A: Any valid unit of code that resolves to a value.

76. **Q:** What is a statement?
A: Instructions that perform an action.

77. **Q:** What are reserved keywords?
A: Words like if, for, function that have special meaning in JavaScript.

78. **Q:** What is eval() used for?
A: To execute a string as JavaScript code (use with caution).

79. **Q:** What does Math.random() return?
A: A random number between 0 and 1.

80. **Q:** What does Math.floor(4.9) return?
A: 4

81. **Q:** What does Math.ceil(4.1) return?
A: 5

82. **Q:** What does Math.round(4.5) return?
A: 5

83. **Q:** What does Math.max(2, 5, 1) return?
A: 5

84. **Q:** What does Math.min(2, 5, 1) return?
A: 1

85. **Q:** How do you generate a random integer from 1 to 10?
A:

```
Math.floor(Math.random() * 10) + 1;
```

86. **Q:** What does typeof null return?
A: "object"

87. **Q:** What is the use of typeof operator?
A: To determine the type of a value.

88. **Q:** What are comments used for?
A: To explain code and make it more readable.

89. **Q:** What is the difference between ++x and x++?
A: ++x increments before returning; x++ returns then increments.

90. **Q:** What is a semicolon used for in JavaScript?
A: To terminate statements (optional but recommended).

91. **Q:** What are escape characters in strings?
A: Special characters like \n, \t, \", etc.

92. **Q:** What is string interpolation?
A: Embedding variables inside strings using template literals.

93. **Q:** What is the difference between slice() and substring()?
A: Both extract string parts, but slice() can accept negative indices.

94. **Q:** What does charAt(0) do?
A: Returns the first character of a string.

95. **Q:** How do you change a string to uppercase?
A: Using .toUpperCase()

96. **Q:** How do you change a string to lowercase?
A: Using .toLowerCase()

97. **Q:** How do you remove whitespace from both ends of a string?
A: Using .trim()

98. **Q:** What is the result of "5" + 5?
A: "55"

99. **Q:** What is the result of "5" - 5?
A: 0

100.Q: What does alert(typeof null) show?
A: "object"

101. **Q:** What are non-primitive data types?
A: Objects, Arrays, Functions.

102. **Q:** What is the difference between primitive and non-primitive data types?
A: Primitive types store single values; non-primitives store collections or objects.

103. **Q:** What is undefined in JavaScript?
A: A variable that has been declared but not assigned a value.

104. **Q:** What is null in JavaScript?
A: A deliberate non-value assigned to a variable.

105. **Q:** What is BigInt?
A: A data type for very large integers beyond the safe range of Number.

106. **Q:** How do you declare a BigInt?
A: Using n at the end of a number: 1234567890123456789012345n

107. **Q:** What is Symbol used for?
A: To create unique identifiers for object properties.

108. **Q:** What is dynamic typing?
A: Variables can hold values of any type, and types can change at runtime.

109. **Q:** How do you check a variable's type?
A: Using typeof.

110. **Q:** What is the output of typeof []?
A: "object"

111. **Q:** What is the output of typeof null?
A: "object"

112. **Q:** What is the output of typeof NaN?
A: "number"

113. **Q:** What is a variable?
A: A named container for storing data.

114. **Q:** What are the three ways to declare a variable in JavaScript?
A: var, let, and const

115. **Q:** What is the difference between var, let, and const?
A: var is function-scoped, let and const are block-scoped; const cannot be reassigned.

116. **Q:** Can you reassign a const variable?
A: No.

117. **Q:** Can a const object's properties be changed?
A: Yes, but the object reference itself can't change.

118. **Q:** Can let be redeclared in the same scope?
A: No.

119. **Q:** Can var be redeclared in the same scope?
A: Yes.

120. **Q:** Is JavaScript statically or dynamically typed?
A: Dynamically typed.

121. **Q:** What is variable hoisting?
A: Variables declared with var are moved to the top of their scope during execution.

122. **Q:** Does let or const support hoisting?
A: They are hoisted but not initialized.

123. **Q:** What is the Temporal Dead Zone (TDZ)?
A: The time between entering the scope and variable initialization with let or const.

124. **Q:** What is block scope?
A: Scope limited to the {} block where the variable is declared.

125. **Q:** What is function scope?
A: Variables are accessible only within the function where they are declared.

126. **Q:** Can you declare a variable without a keyword?
A: Yes, but it becomes global (not recommended).

127. **Q:** How do you declare multiple variables in one line?
A: let a = 1, b = 2, c = 3;

128. **Q:** What is the best practice for declaring variables?
A: Use const by default, let if value will change, avoid var.

129. **Q:** What is the default value of an uninitialized variable?
A: undefined

130. **Q:** Can a variable hold different types at different times?
A: Yes.

131. **Q:** What happens if you use a variable before declaring it with var?
A: It is undefined.

132. **Q:** What happens if you use a variable before declaring it with let or const?
A: ReferenceError.

133. **Q:** Is NaN equal to NaN?
A: No.

134. **Q:** How do you check for NaN properly?
  A: Using Number.isNaN().

135. **Q:** What is the result of 0 / 0?
  A: NaN

136. **Q:** What is the result of 10 / 0?
  A: Infinity

137. **Q:** What is Infinity in JavaScript?
  A: A special value greater than any number.

138. **Q:** How do you get negative infinity?
  A: -Infinity

139. **Q:** What is a literal in JavaScript?
  A: A fixed value like 10, "Hello", or true.

140. **Q:** Can a string be empty?
  A: Yes, using "".

141. **Q:** How do you find the length of a variable's value if it's a string?
  A: Use .length property.

142. **Q:** What happens if you access an undeclared variable?
  A: ReferenceError.

143. **Q:** What is the difference between typeof undefined and typeof undeclaredVar?
  A: Both return "undefined", but accessing undeclaredVar directly throws an error.

144. **Q:** What is implicit type conversion?
  A: Automatic type change done by JavaScript.

145. **Q:** What is explicit type conversion?
  A: Manual type change using functions like Number(), String().

146. **Q:** What is truthy in JavaScript?
  A: Any value that evaluates to true in a boolean context.

147. **Q:** What is falsy in JavaScript?
  A: Values like 0, "", false, null, undefined, NaN.

148. **Q:** Is "false" (a string) truthy or falsy?
  A: Truthy.

149. **Q:** Is [] truthy or falsy?
  A: Truthy.

151. **Q:** What are JavaScript operators?
  A: Symbols used to perform operations on operands (e.g., +, -, ==, &&).

152. **Q:** List some arithmetic operators.
  A: +, -, *, /, %, ** (exponentiation).

153. **Q:** What does ** do?
  A: Performs exponentiation: 2 ** 3 = 8.

154. **Q:** What is the modulus operator?
  A: % – returns remainder of division.

155. **Q:** What is the result of 5 % 2?
  A: 1

156. **Q:** What are assignment operators?
  A: Used to assign values: =, +=, -=, *=, /=, %=, etc.

157. **Q:** What does x += 5 mean?
  A: x = x + 5

158. **Q:** What are comparison operators?
  A: ==, !=, ===, !==, >, <, >=, <=

159. **Q:** What's the difference between == and ===?
  A: == compares values with type conversion; === compares both value and type.

160. **Q:** What is the output of 5 == "5"?
   A: true
161. **Q:** What is the output of 5 === "5"?
   A: false
162. **Q:** What is the difference between != and !==?
   A: != compares value (not type), !== compares both value and type.
163. **Q:** What are logical operators?
   A: && (AND), || (OR), ! (NOT)
164. **Q:** What does true && false return?
   A: false
165. **Q:** What does true || false return?
   A: true
166. **Q:** What does !true return?
   A: false
167. **Q:** What is a ternary operator?
   A: A shorthand for if-else: condition ? expr1 : expr2
168. **Q:** Give an example of the ternary operator.
   A: let result = age >= 18 ? "Adult" : "Minor"
169. **Q:** What is operator precedence?
   A: The order in which operators are evaluated.
170. **Q:** Which has higher precedence: * or +?
   A: *
171. **Q:** What is associativity in operators?
   A: The direction in which operators with the same precedence are evaluated.
172. **Q:** What is the associativity of = operator?
   A: Right-to-left

173. **Q:** What is the output of 3 + 4 * 2?
A: 11 (multiplication has higher precedence)

174. **Q:** What is a unary operator?
A: An operator with a single operand (e.g., typeof, !, ++, --).

175. **Q:** What is the output of typeof 42?
A: "number"

176. **Q:** What are increment/decrement operators?
A: ++ increases, -- decreases a value.

177. **Q:** What is the difference between ++x and x++?
A: ++x increments before returning, x++ returns before incrementing.

178. **Q:** What does null == undefined return?
A: true

179. **Q:** What does null === undefined return?
A: false

180. **Q:** What are bitwise operators?
A: Operate on binary values: &, |, ^, ~, <<, >>, >>>

181. **Q:** What does 5 & 1 return?
A: 1

182. **Q:** What does 5 | 1 return?
A: 5

183. **Q:** What is the result of typeof NaN?
A: "number"

184. **Q:** What is the comma operator?
A: Evaluates multiple expressions and returns the last:
let x = (1, 2, 3); // x = 3

185. **Q:** Can you chain assignments?
A: Yes: a = b = c = 10

186. **Q:** What does !!value do?
   A: Converts value to Boolean.
187. **Q:** What is optional chaining (?.) used for?
   A: To safely access deeply nested properties.
188. **Q:** What does obj?.prop return if obj is undefined?
   A: undefined, instead of throwing an error.
189. **Q:** What is nullish coalescing (??)?
   A: Returns right-hand value if left-hand is null or undefined.
190. **Q:** What is the result of null ?? "default"?
   A: "default"
191. **Q:** What is the result of 0 ?? "default"?
   A: 0 (since 0 is not null or undefined)
192. **Q:** What is the difference between || and ???
   A: || returns the first truthy value, ?? returns the first defined value.
193. **Q:** What is short-circuiting?
   A: When the result is determined without evaluating all expressions.
194. **Q:** What does x && y return if x is falsy?
   A: x
195. **Q:** What does x || y return if x is truthy?
   A: x
196. **Q:** What is a compound expression?
   A: A combination of multiple expressions.
197. **Q:** Can operators be overloaded in JavaScript?
   A: No.
198. **Q:** What does 1 + "2" return?
   A: "12" (string concatenation)

199. **Q:** What does "2" - 1 return?
 A: 1 (type coercion to number)
200. **Q:** What does "2" + 1 + 3 return?
 A: "213"
201. **Q:** What are control structures in JavaScript?
 A: They control the flow of code execution (e.g., if, for, while, switch).
202. **Q:** What is an if statement?
 A: A conditional statement that runs code if a condition is true.
203. **Q:** What is the syntax of an if statement?
 A:

```
if (condition) {
  // code
}
```

204. **Q:** What is an if-else statement?
 A: Adds an alternative block if the condition is false.
205. **Q:** What is an else if statement?
 A: Adds more conditions after the first if.
206. **Q:** Can if-else statements be nested?
 A: Yes.
207. **Q:** What is a ternary operator used for?
 A: For short if-else decisions in a single line.
208. **Q:** What is a switch statement?
 A: Used to select one of many code blocks based on a value.
209. **Q:** What keyword ends a case block in switch?
 A: break

210. **Q:** What happens if break is missing in a switch case?
A: The next case executes (fall-through).

211. **Q:** What is a default case in switch?
A: Executes if no case matches.

212. **Q:** Can switch evaluate expressions?
A: Yes, but it matches exact values.

213. **Q:** What are loops in JavaScript?
A: Structures that repeat code multiple times.

214. **Q:** What is a for loop?
A: A loop with initialization, condition, and increment.

215. **Q:** What is the syntax of a for loop?
A:

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

216. **Q:** What is a while loop?
A: Repeats code while a condition is true.

217. **Q:** What is the syntax of a while loop?
A:

```
while (condition) {
  // code
}
```

218. **Q:** What is a do-while loop?
A: Executes code at least once, then repeats while condition is true.

219. **Q:** How is do-while different from while?
A: do-while always runs at least once.

220. **Q:** What is an infinite loop?
A: A loop that never ends due to a condition that always remains true.

221. **Q:** How do you exit a loop early?
A: Using break

222. **Q:** How do you skip the current iteration in a loop?
A: Using continue

223. **Q:** What is the output of:

```
for (let i = 0; i < 3; i++) {
  console.log(i);
}
```

A: 0, 1, 2

224. **Q:** Can you nest loops?
A: Yes.

225. **Q:** What is the use of labels in loops?
A: To identify and break from specific outer loops.

226. **Q:** Can you use break outside a loop or switch?
A: No, it causes a syntax error.

227. **Q:** Can loops be used with arrays?
A: Yes, especially for, for-of, forEach.

228. **Q:** What is for...of used for?
A: Iterates over iterable objects like arrays or strings.

229. **Q:** What is for...in used for?
A: Iterates over object keys.

230. **Q:** What is the difference between for...in and for...of?
A: for...in gives keys, for...of gives values.

231. **Q:** What is the output of:

```
for (let char of "Hi") {
  console.log(char);
}
```

A: H, i

232. **Q:** What is the output of:

```
const obj = {a: 1, b: 2};
for (let key in obj) {
  console.log(key);
}
```

A: a, b

233. **Q:** Can you use continue in a while loop?
    A: Yes.
234. **Q:** Can switch cases be combined?
    A: Yes:

```
case 1:
case 2:
  // code
  break;
```

235. **Q:** What is a common use case for continue?
    A: Skipping even numbers, filtering, etc.
236. **Q:** Can you put if statements inside loops?
    A: Yes.
237. **Q:** Is switch faster than if-else?
    A: Sometimes, but both depend on context.
238. **Q:** What happens if you use const in a loop?
    A: It works only if redeclared inside the loop block.

239. **Q:** What happens if the loop condition is always false?
  A: The loop doesn’t run.

240. **Q:** Can for loop run backward?
  A: Yes:

```
for (let i = 5; i >= 0; i--) { }
```

241. **Q:** Can loops be used with objects?
  A: for...in is commonly used.

242. **Q:** What is the use of labels with nested loops?
  A: To break out of the outer loop from inside an inner loop.

243. **Q:** Can while be used to wait?
  A: Not effectively; it blocks the thread.

244. **Q:** Should you use infinite loops in JavaScript?
  A: Only with exit conditions; otherwise, it freezes the page.

245. **Q:** Can a while loop be converted to a for loop?
  A: Yes, if properly structured.

246. **Q:** Are for loops synchronous or asynchronous?
  A: Synchronous.

247. **Q:** What is the scope of a variable declared inside a loop?
  A: Block scope if declared with let or const.

248. **Q:** Can you break from a nested forEach() loop using break?
  A: No, use regular loops or exceptions.

249. **Q:** Can a switch case check for ranges?
  A: No, use if-else instead.

250. **Q:** What happens if a loop variable is not updated?
  A: It may cause an infinite loop.

301. **Q:** What is a function in JavaScript?
  A: A block of reusable code that performs a specific task.

302. **Q:** How do you define a function?
A:

```
function greet() {
  console.log("Hello!");
}
```

303. **Q:** How do you call a function?
A: By using its name followed by parentheses: greet();
304. **Q:** What are parameters in a function?
A: Placeholders in the function definition to receive values.
305. **Q:** What are arguments in a function?
A: Actual values passed to a function when calling it.
306. **Q:** What is the return statement?
A: It exits a function and optionally returns a value.
307. **Q:** What is the output of:

```
function add(a, b) {
  return a + b;
}
console.log(add(2, 3));
```

A: 5

308. **Q:** Can a function return multiple values?
A: No directly, but you can return an array or object.
309. **Q:** What is a function expression?
A: Assigning a function to a variable:

```
const greet = function() {
  return "Hello";
};
```

310. **Q:** What is an anonymous function?
A: A function without a name, often used in expressions.

311. **Q:** What is an arrow function?
A: A shorter syntax for writing functions:

```
const greet = () => "Hello";
```

312. **Q:** What is the syntax for an arrow function with parameters?
A:

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

313. **Q:** Can arrow functions have a return keyword?
A: Yes, if using a block body:

```
(a, b) => { return a + b; }
```

314. **Q:** Can arrow functions be used as constructors?
A: No, they cannot be used with new.

315. **Q:** What is a callback function?
A: A function passed as an argument to another function.

316. **Q:** Give an example of a callback function.
A:

```
setTimeout(function() {
  console.log("Callback called");
}, 1000);
```

317. **Q:** What is a higher-order function?
A: A function that takes another function as an argument or returns one.

318. **Q:** Are functions first-class objects in JS?
A: Yes, they can be passed and returned like variables.

319. **Q:** What is function hoisting?
    A: Function declarations are moved to the top during compilation.
320. **Q:** Are function expressions hoisted?
    A: No, only declarations are hoisted.
321. **Q:** What is the difference between function declaration and expression?
    A: Declaration is hoisted; expression is not.
322. **Q:** Can you assign a function to a variable?
    A: Yes.
323. **Q:** Can you store functions in an array?
    A: Yes.
324. **Q:** Can functions be nested?
    A: Yes.
325. **Q:** What is a recursive function?
    A: A function that calls itself.
326. **Q:** Give an example of recursion.
    A:

```
function factorial(n) {
  if (n <= 1) return 1;
  return n * factorial(n - 1);
}
```

327. **Q:** What is the base case in recursion?
    A: The condition where recursion stops.
328. **Q:** What is the stack overflow error in recursion?
    A: It occurs when a recursive function calls itself too many times without a base case.

329. **Q:** What are default parameters?
A: Parameters that have default values if not passed:

```
function greet(name = "Guest") { }
```

330. **Q:** What is the arguments object?
A: An array-like object containing all passed arguments.

331. **Q:** Do arrow functions have arguments object?
A: No.

332. **Q:** Can you spread arguments using ...?
A: Yes, it’s called the rest parameter:

```
function sum(...numbers) { }
```

333. **Q:** What is the use of rest parameters?
A: To accept unlimited arguments as an array.

334. **Q:** Can functions be passed as arguments?
A: Yes.

335. **Q:** Can functions return other functions?
A: Yes.

336. **Q:** What is function scope?
A: Variables defined inside a function are only accessible there.

337. **Q:** What is block scope?
A: Variables with let/const inside {} are accessible only within that block.

338. **Q:** Do functions create their own scope?
A: Yes.

339. **Q:** Can you use functions inside conditionals?
A: Yes.

340. **Q:** Can you call a function inside another function?
A: Yes.
341. **Q:** What does IIFE stand for?
A: Immediately Invoked Function Expression.
342. **Q:** What is the syntax of IIFE?
A:

```
(function() {
  console.log("Run immediately");
})();
```

343. **Q:** Why use IIFE?
A: To create private scopes and avoid polluting global scope.
344. **Q:** Can IIFE be arrow functions?
A: Yes.
345. **Q:** Can IIFE return values?
A: Yes:

```
let result = (() => 5 + 2)();
```

346. **Q:** What is memoization?
A: Caching function results to improve performance.
347. **Q:** What is function currying?
A: Converting a function with multiple arguments into a sequence of functions.
348. **Q:** Give an example of currying.
A:

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

349. **Q:** What are pure functions?
    A: Functions with no side effects and consistent output for the same input.
350. **Q:** Why are pure functions useful?
    A: They are predictable and easier to test.
351. **Q:** What is an impure function?
    A: A function that modifies external variables or has side effects.
352. **Q:** What is the benefit of named functions?
    A: Easier debugging and recursion.
353. **Q:** Can you redefine a function?
    A: Yes, last definition overrides the earlier ones.
354. **Q:** Can a function modify a global variable?
    A: Yes, if it's accessed directly.
355. **Q:** What is the effect of missing return in a function?
    A: Returns undefined by default.
356. **Q:** Is JavaScript pass-by-value or pass-by-reference?
    A: Primitives are passed by value, objects by reference.
357. **Q:** Can you use arrow functions inside objects?
    A: Yes, but they don't have their own this.
358. **Q:** Can arrow functions be methods?
    A: Not recommended, due to lack of this.
359. **Q:** What's the difference between normal and arrow function this?
    A: Normal functions bind this dynamically; arrow functions use lexical this.

360. **Q:** What is lexical scope?
A: Scope determined by the location of function definition.

361. **Q:** What is the difference between static and dynamic scoping?
A: JavaScript uses lexical (static) scoping.

362. **Q:** Can functions be asynchronous?
A: Yes, using async keyword.

363. **Q:** What does async keyword do?
A: Allows use of await inside a function and returns a promise.

364. **Q:** What does await do?
A: Pauses execution until a promise resolves.

365. **Q:** Can normal functions use await?
A: No, only inside async functions.

366. **Q:** What is the return type of an async function?
A: A Promise.

367. **Q:** What happens if an async function throws an error?
A: It returns a rejected promise.

368. **Q:** Can you await multiple promises?
A: Yes, use Promise.all() with await.

369. **Q:** What is a generator function?
A: A special function that can be paused and resumed using function*.

370. **Q:** What keyword resumes a generator?
A: next()

371. **Q:** Can generator functions be asynchronous?
A: Yes, use async function*.

372. **Q:** What is a closure?
A: A function that retains access to its outer scope even after the outer function has returned.

373. **Q:** Give an example of a closure.
A:

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

374. **Q:** What is the use of closures?
A: To create private variables or encapsulation.

375. **Q:** Are closures created with arrow functions?
A: Yes.

376. **Q:** Can closures cause memory leaks?
A: If not managed carefully, yes.

377. **Q:** What is function chaining?
A: Returning this or another function to allow consecutive calls.

378. **Q:** Can you use this inside functions?
A: Yes, depends on how the function is called.

379. **Q:** Can you bind a function’s this manually?
A: Yes, using .bind(), .call(), or .apply().

380. **Q:** What does bind() do?
A: Creates a new function with a fixed this context.

381. **Q:** What is the difference between call() and apply()?
A: call() takes arguments normally; apply() takes an array.

382. **Q:** What is method chaining?
A: Calling multiple methods on the same object in a single statement.

383. **Q:** What does .toString() do for a function?
A: Returns the source code of the function as a string.

384. **Q:** Can you use new Function() constructor?
A: Yes, but it's not recommended due to security and performance issues.

385. **Q:** Can functions be used with classes?
A: Yes, as methods or static functions.

386. **Q:** Can functions be async generators?
A: Yes: async function*.

387. **Q:** What is a tail call?
A: A function call that happens as the last action in a function.

388. **Q:** Does JavaScript support tail call optimization?
A: Some engines do, but not universally supported.

389. **Q:** What's the purpose of the function constructor?
A: Dynamically creates functions:
let fn = new Function('a', 'b', 'return a + b');

390. **Q:** Is using function constructor safe?
A: No, it's like eval() and can be unsafe.

391. **Q:** What is a method?
A: A function that is a property of an object.

392. **Q:** Can you delete a function?
A: Only if it's a property of an object or variable.

393. **Q:** Can you clone a function?
 A: Not directly, but you can copy its reference.

394. **Q:** Can you stringify a function?
 A: Yes, using .toString().

395. **Q:** What is the difference between static and instance methods?
 A: Static belongs to the class, instance to the object.

396. **Q:** Can functions be stored in objects?
 A: Yes.

397. **Q:** Can functions have properties?
 A: Yes, because functions are objects.

398. **Q:** What is the arity of a function?
 A: The number of arguments a function expects.

399. **Q:** How to find the arity of a function?
 A: Using function.length

400. **Q:** What is the use of .name property of a function?
 A: Returns the name of the function.

401. **Q:** What is an object in JavaScript?
 A: A collection of key-value pairs.

402. **Q:** How do you create an object?
 A:

```
let obj = { name: "John", age: 30 };
```

403. **Q:** How do you access object properties?
 A: Dot or bracket notation: obj.name or obj["name"]

404. **Q:** How do you add a new property to an object?
 A:

```
obj.city = "New York";
```

405. **Q:** How do you delete a property from an object?
A:

```
delete obj.age;
```

406. **Q:** How do you check if a property exists?
A:

```
"name" in obj
```

407. **Q:** What is Object.keys(obj)?
A: Returns an array of keys.
408. **Q:** What is Object.values(obj)?
A: Returns an array of values.
409. **Q:** What is Object.entries(obj)?
A: Returns an array of key-value pairs.
410. **Q:** What does typeof obj return for an object?
A: "object"
411. **Q:** Can object keys be numbers?
A: Yes, but they are stored as strings.
412. **Q:** What is a method in an object?
A: A function stored as a property.
413. **Q:** How do you define a method?
A:

```
let user = {
  greet: function() {
    return "Hello";
  }
};
```

414. **Q:** What is shorthand for method definition?
A:

```
greet() {
  return "Hello";
}
```

415. **Q:** What is this inside an object method?
   A: Refers to the object itself.
416. **Q:** What does Object.assign() do?
   A: Copies properties from one or more objects to a target object.
417. **Q:** What is object destructuring?
   A:

```
let {name, age} = user;
```

418. **Q:** Can you nest objects?
   A: Yes, objects can have other objects as values.
419. **Q:** How do you loop through object properties?
   A: Using for...in loop.
420. **Q:** Can you clone an object?
   A: Use Object.assign() or spread syntax:

```
let clone = {...obj};
```

421. **Q:** What is the difference between shallow and deep copy?
   A: Shallow copy duplicates top-level only; deep copy includes nested structures.
422. **Q:** How to perform a deep copy?
   A: Use JSON.parse(JSON.stringify(obj)) or structuredClone.
423. **Q:** What is Object.freeze()?
   A: Makes an object immutable.

424. **Q:** What is Object.seal()?
A: Prevents adding/removing properties but allows editing existing ones.
425. **Q:** What is a constructor function?
A: A function used to create multiple similar objects.
426. **Q:** How to create an object using a constructor?
A:

```
function User(name) {
 this.name = name;
}
let u = new User("John");
```

427. **Q:** What is new keyword?
A: Creates an instance of an object from a constructor.
428. **Q:** What does instanceof check?
A: If an object is created by a specific constructor.
429. **Q:** What is hasOwnProperty()?
A: Checks if a property exists directly on the object (not inherited).
430. **Q:** Can objects be compared directly?
A: Only by reference, not value.
431. **Q:** What is Object.create()?
A: Creates a new object with the specified prototype.
432. **Q:** What is the prototype of an object?
A: An object from which it inherits properties/methods.
433. **Q:** What is __proto__?
A: A property that refers to an object's prototype.

434. **Q:** Can you create objects with no prototype?
A: Yes: Object.create(null)

435. **Q:** How do you get all property names of an object?
A: Use Object.getOwnPropertyNames(obj)

436. **Q:** What is the difference between primitive and object types?
A: Primitives hold values, objects hold references.

437. **Q:** Are arrays objects?
A: Yes.

438. **Q:** Is null an object?
A: Yes, typeof null returns "object" (quirk of JavaScript).

439. **Q:** How to merge two objects?
A:

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

440. **Q:** What happens if both objects have same key?
A: The later key overwrites the former.

441. **Q:** Can object properties have functions?
A: Yes, they are called methods.

442. **Q:** Can object keys be symbols?
A: Yes.

443. **Q:** What is a symbol?
A: A unique and immutable primitive used as object keys.

444. **Q:** How to get symbol properties?
A: Use Object.getOwnPropertySymbols(obj)

445. **Q:** What is a computed property name?
A: Dynamically created property key:

```
let key = "name";
let obj = { [key]: "John" };
```

446. **Q:** What is object spread syntax?
   A: {...obj} – copies properties.
447. **Q:** What is a property descriptor?
   A: An object that describes attributes of a property (writable, enumerable, configurable).
448. **Q:** How to define properties with descriptors?
   A:

```
Object.defineProperty(obj, 'key', { writable: false });
```

449. **Q:** What is enumerable?
   A: Determines if the property appears in loops.
450. **Q:** What is configurable?
   A: Determines if a property can be deleted or changed.
451. **Q:** What happens if you assign an object to another variable?
   A: Both variables refer to the same object in memory.
452. **Q:** How can you prevent modification of an object?
   A: Use Object.freeze(obj)
453. **Q:** How do you make a property read-only?
   A: Using Object.defineProperty() with writable: false.
454. **Q:** What is a getter in JavaScript?
   A: A method that gets the value of a property.
455. **Q:** What is a setter in JavaScript?
   A: A method that sets the value of a property.
456. **Q:** How do you define a getter?
   A:

```
let obj = {
  get name() {
    return "John";
  }
};
```

457. **Q:** How do you define a setter?
    A:

```
let obj = {
  set name(val) {
    this._name = val;
  }
};
```

458. **Q:** Can getters and setters coexist for the same property?
    A: Yes.
459. **Q:** Can you define multiple properties with Object.defineProperties()?
    A: Yes, to set multiple descriptors at once.
460. **Q:** What does Object.getOwnPropertyDescriptor() do?
    A: Returns the property descriptor for a specific property.
461. **Q:** What does Object.getPrototypeOf(obj) do?
    A: Returns the prototype of the given object.
462. **Q:** What does Object.setPrototypeOf(obj, proto) do?
    A: Sets a new prototype for an object.
463. **Q:** What are inherited properties?
    A: Properties received from the prototype chain.
464. **Q:** How to check if a property is inherited?
    A: Use obj.hasOwnProperty("key")
465. **Q:** What is toString() method of an object?
    A: Converts the object to a string.

466. **Q:** Can you override toString()?
A: Yes, by defining a custom method.

467. **Q:** What is valueOf() in objects?
A: Returns the primitive value of the object.

468. **Q:** What is object coercion?
A: Converting an object to a string or number.

469. **Q:** What happens if you stringify an object with circular reference?
A: JSON.stringify() will throw an error.

470. **Q:** How do you handle circular references?
A: Use a custom replacer or third-party libraries like flatted.

471. **Q:** How to get JSON from an object?
A: Use JSON.stringify(obj)

472. **Q:** How to parse JSON into an object?
A: Use JSON.parse(jsonStr)

473. **Q:** What is the purpose of a factory function?
A: To create and return objects without using new.

474. **Q:** What is an object initializer?
A: An expression using {} to create objects.

475. **Q:** Can you loop through an object with forEach()?
A: No, forEach() is for arrays. Use Object.keys(obj).forEach() instead.

476. **Q:** What does Object.is(a, b) do?
A: Checks if two values are the same (like === but more accurate with NaN and -0).

477. **Q:** Can object methods use arrow functions?
A: Yes, but they don’t bind this to the object.

478. **Q:** When should you avoid arrow functions in objects?
A: When using this inside methods.

479. **Q:** What’s the output of {a: 1} == {a: 1}?
A: false – different object references.

480. **Q:** What is the result of JSON.stringify({a: undefined})?
A: '{}' – undefined is excluded.

481. **Q:** What is a plain object?
A: An object created with {} or new Object().

482. **Q:** What is a dictionary in JavaScript?
A: Just a regular object used for key-value storage.

483. **Q:** How do you safely access nested object properties?
A: Use optional chaining: obj?.user?.name

484. **Q:** What is optional chaining (?.)?
A: It prevents errors when accessing properties of null or undefined.

485. **Q:** How to set a default value with optional chaining?
A:

```
let name = obj?.user?.name ?? "Guest";
```

486. **Q:** What does ?? (nullish coalescing) do?
A: Returns the right operand only if the left is null or undefined.

487. **Q:** What is a dynamic property in an object?
A: A property whose key is evaluated at runtime.

488. **Q:** How to loop object values only?
A:

```
Object.values(obj).forEach(v => console.log(v));
```

489. **Q:** What is the main drawback of for...in loop?
A: It also iterates over inherited properties.

490. **Q:** How to exclude inherited properties while looping?
A: Use obj.hasOwnProperty(key) inside for...in.

491. **Q:** Are objects passed by reference or value?
A: By reference.

492. **Q:** How do you make an object iterable?
A: By implementing [Symbol.iterator]() method.

493. **Q:** Can you store functions in object properties?
A: Yes.

494. **Q:** Can an object be empty?
A: Yes: {}

495. **Q:** How to check if an object is empty?
A:

Object.keys(obj).length === 0

496. **Q:** What is Object.fromEntries()?
A: Converts key-value pair arrays into an object.

497. **Q:** What is the use of Object.entries() + map()?
A: For transforming object key-value pairs.

498. **Q:** What does Object.hasOwn() do?
A: Checks if an object has a property (safer than hasOwnProperty()).

499. **Q:** Can you spread objects inside arrays?
A: Yes: [...Object.values(obj)]

500. **Q:** Can you nest arrays inside objects?
A: Yes.

501. **Q:** What is an array in JavaScript?
A: A list-like object used to store multiple values.

502. **Q:** How do you create an array?
A:

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

503. **Q:** How to access array elements?
A: Using index: arr[0]

504. **Q:** What is the index of the first element in an array?
A: 0

505. **Q:** How to find the length of an array?
A: arr.length

506. **Q:** How do you add an element at the end of an array?
A: arr.push(value)

507. **Q:** How to remove the last element?
A: arr.pop()

508. **Q:** How to add an element at the beginning?
A: arr.unshift(value)

509. **Q:** How to remove the first element?
A: arr.shift()

510. **Q:** How to check if a variable is an array?
A:

```
Array.isArray(arr)
```

511. **Q:** What does arr.includes(value) do?
A: Returns true if the array contains the value.

512. **Q:** What is indexOf() used for?
A: Returns the first index of a value, or -1 if not found.

513. **Q:** What is lastIndexOf()?
   A: Returns the last index of a value.
514. **Q:** How to join array elements into a string?
   A: arr.join(", ")
515. **Q:** What does arr.toString() do?
   A: Converts the array to a comma-separated string.
516. **Q:** How do you reverse an array?
   A: arr.reverse()
517. **Q:** How to sort an array?
   A: arr.sort()
518. **Q:** What is the issue with sort() on numbers?
   A: It sorts alphabetically unless a compare function is used.
519. **Q:** How to sort numerically?
   A:

```
arr.sort((a, b) => a - b)
```

520. **Q:** What does splice() do?
   A: Adds/removes items at a specific index.
521. **Q:** What does slice() do?
   A: Returns a shallow copy of a portion of an array.
522. **Q:** What is the difference between slice() and splice()?
   A: slice() doesn't modify the original array; splice() does.
523. **Q:** How to clone an array?
   A: Use slice() or spread: [...arr]
524. **Q:** How to concatenate arrays?
   A: arr1.concat(arr2)
525. **Q:** Can arrays store different data types?
   A: Yes.

526. **Q:** How do you loop through an array?
A: With for, for...of, forEach, or array methods.

527. **Q:** What is forEach()?
A: Executes a function for each array element.

528. **Q:** Can forEach() break the loop?
A: No, use a regular loop for that.

529. **Q:** What is map()?
A: Creates a new array by transforming each element.

530. **Q:** What is filter()?
A: Creates a new array with elements that pass a condition.

531. **Q:** What is reduce()?
A: Applies a function to reduce the array to a single value.

532. **Q:** What is find()?
A: Returns the first element that matches a condition.

533. **Q:** What is findIndex()?
A: Returns the index of the first matching element.

534. **Q:** What is every()?
A: Returns true if all elements match a condition.

535. **Q:** What is some()?
A: Returns true if at least one element matches a condition.

536. **Q:** What does flat() do?
A: Flattens nested arrays.

537. **Q:** What does flatMap() do?
A: Maps and flattens the result by one level.

538. **Q:** What is Array.from()?
A: Converts array-like or iterable objects to an array.

539. **Q:** What is Array.of()?
   A: Creates an array from given arguments.
540. **Q:** What is fill()?
   A: Fills all or part of the array with a static value.
541. **Q:** What is copyWithin()?
   A: Copies part of the array to another location in the same array.
542. **Q:** What is the spread operator in arrays?
   A: Expands elements: [...arr]
543. **Q:** What does arr.length = 0 do?
   A: Clears the array.
544. **Q:** Can arrays be nested?
   A: Yes, arrays can contain other arrays.
545. **Q:** What is a multidimensional array?
   A: An array of arrays (e.g., [[1,2],[3,4]]).
546. **Q:** How to flatten a nested array deeply?
   A:

```
arr.flat(Infinity)
```

547. **Q:** What is a sparse array?
   A: An array with missing indices.
548. **Q:** What happens if you access an index that doesn't exist?
   A: It returns undefined.
549. **Q:** What is the best way to loop both index and value?
   A:

```
arr.forEach((value, index) => {});
```

550. **Q:** Can you use for...in for arrays?
   A: Not recommended — use for...of or forEach instead.

551. **Q:** How do you remove duplicates from an array?
A:

```
[...new Set(arr)]
```

552. **Q:** How to get the max value in an array?
A:

```
Math.max(...arr)
```

553. **Q:** How to get the min value in an array?
A:

```
Math.min(...arr)
```

554. **Q:** How to shuffle an array randomly?
A: Use the Fisher-Yates algorithm.

555. **Q:** What is array destructuring?
A: Assigning array elements to variables:

```
let [a, b] = [1, 2];
```

556. **Q:** Can you destructure from nested arrays?
A: Yes:

```
let [a, [b, c]] = [1, [2, 3]];
```

557. **Q:** What is the rest operator in destructuring?
A:

```
let [a, ...rest] = [1, 2, 3];
```

558. **Q:** Can you skip values during destructuring?
A: Yes:

```
let [,,c] = [1, 2, 3];
```

559. **Q:** How to check if an array is empty?
A: arr.length === 0

560. **Q:** How to convert a string to an array?
A:

```
str.split("")
```

561. **Q:** How to convert an array to a string?
  A:

```
arr.join("")
```

562. **Q:** How to create an array with N elements?
  A:

```
Array(n).fill(0)
```

563. **Q:** How to map array to uppercase strings?
  A:

```
arr.map(str => str.toUpperCase())
```

564. **Q:** How to filter out falsy values?
  A:

```
arr.filter(Boolean)
```

565. **Q:** How to remove specific value from array?
  A:

```
arr.filter(val => val !== target)
```

566. **Q:** How to sum numbers in an array?
  A:

```
arr.reduce((a, b) => a + b, 0)
```

567. **Q:** What is the use of Array.prototype?
  A: To add new methods to all arrays.

568. **Q:** What's the difference between map() and forEach()?
  A: map() returns a new array, forEach() doesn't.

569. **Q:** What is Array.prototype.every.call()?
  A: Applies every() to array-like objects.

570. **Q:** Can you chain array methods?
  A: Yes:

```
arr.filter(...).map(...).reduce(...)
```

571. **Q:** How to find all indices of a value?
 A: Use reduce() or loop with indexOf repeatedly.
572. **Q:** What is an array-like object?
 A: An object with indexed elements and length, like arguments.
573. **Q:** How to convert arguments to array?
 A:

```
Array.from(arguments)
```

574. **Q:** What's the output of [1,2] + [3,4]?
 A: "1,23,4" – string concatenation.
575. **Q:** What is the output of [,,,]?
 A: A sparse array with 3 empty slots.
576. **Q:** Can an array have undefined values?
 A: Yes.
577. **Q:** What happens if you delete an element using delete arr[1]?
 A: Leaves a hole (sparse array), doesn't reduce length.
578. **Q:** Best way to remove an element by index?
 A:

```
arr.splice(index, 1)
```

579. **Q:** Is typeof arr equal to "array"?
 A: No, it returns "object".
580. **Q:** Why is Array.isArray() better than typeof?
 A: Because it correctly checks for arrays.
581. **Q:** How to fill array with sequential numbers?
 A:

```
Array.from({length: 5}, (_, i) => i + 1)
```

582. **Q:** Can you sort strings in descending order?
    A:

```
arr.sort().reverse()
```

583. **Q:** Can you sort numbers descending?
    A:

```
arr.sort((a, b) => b - a)
```

584. **Q:** What does arr.at(-1) do?
    A: Gets the last item (negative indexing).
585. **Q:** What is the spread operator used for in arrays?
    A: Copy, merge, expand.
586. **Q:** How to check if all elements are numbers?
    A:

```
arr.every(e => typeof e === "number")
```

587. **Q:** How to compare two arrays for equality?
    A: Compare lengths + all elements with every().
588. **Q:** Why [] == [] is false?
    A: Different references.
589. **Q:** Can you use push() on an empty array?
    A: Yes.
590. **Q:** Can you use map() on an empty array?
    A: No, it skips holes in sparse arrays.
591. **Q:** Can arrays have mixed types?
    A: Yes.
592. **Q:** What does arr.length = 100 do?
    A: Sets the length, adds empty slots.

593. **Q:** How to remove last N elements?
   A:

```
arr.splice(-N)
```

594. **Q:** How to remove first N elements?
   A:

```
arr.splice(0, N)
```

595. **Q:** Can push() return anything?
   A: Yes, the new length.
596. **Q:** What happens if you use arr[10] = "x"?
   A: It creates empty slots from index 0–9 if they don't exist.
597. **Q:** Can you assign properties to an array like arr.name = "x"?
   A: Yes, but it doesn't affect array elements.
598. **Q:** Is array.length always equal to number of elements?
   A: No, it can include empty slots.
599. **Q:** What is the time complexity of push()?
   A: O(1)
600. **Q:** What is the time complexity of splice()?
   A: O(n) in worst case.
601. **Q:** What is a function in JavaScript?
   A: A reusable block of code that performs a task.
602. **Q:** How do you define a function?
   A:

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

603. **Q:** How do you call a function?
   A: Using its name and parentheses: greet();
604. **Q:** What are parameters?
   A: Variables listed in the function definition.

605. **Q:** What are arguments?
   A: Values passed to a function when called.
606. **Q:** What is the return statement?
   A: It ends a function and returns a value.
607. **Q:** Can functions return nothing?
   A: Yes, if there's no return, they return undefined.
608. **Q:** What is a function expression?
   A:

```
const add = function(x, y) { return x + y; };
```

609. **Q:** What is a named function expression?
   A: A function expression with a name:

```
const fn = function named() {};
```

610. **Q:** What is a function declaration?
   A:

```
function sayHi() {}
```

611. **Q:** What’s the difference between function declaration and expression?
   A: Declarations are hoisted; expressions are not.
612. **Q:** What is hoisting in functions?
   A: Function declarations are moved to the top of their scope.
613. **Q:** Can functions be assigned to variables?
   A: Yes.
614. **Q:** Can functions be passed as arguments?
   A: Yes, they're first-class objects.
615. **Q:** Can functions be returned from other functions?
   A: Yes.

616. **Q:** What is a higher-order function?
A: A function that takes or returns another function.

617. **Q:** What is a callback function?
A: A function passed to another function to be called later.

618. **Q:** What is an anonymous function?
A: A function without a name.

619. **Q:** Can a function call itself?
A: Yes, it's called recursion.

620. **Q:** What is recursion?
A: When a function calls itself.

621. **Q:** When should you use recursion?
A: For problems that can be broken into similar sub-problems (e.g., factorial, tree traversal).

622. **Q:** What is the base case in recursion?
A: The condition under which recursion stops.

623. **Q:** What happens if there's no base case in recursion?
A: Stack overflow (infinite loop).

624. **Q:** Can functions have default parameter values?
A: Yes:

```
function greet(name = "Guest") {}
```

625. **Q:** How to access all arguments in a function?
A: Use arguments object (non-arrow functions only).

626. **Q:** What is an arrow function?
A: A short syntax for writing functions.

627. **Q:** How do you write an arrow function?
A:

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

628. **Q:** Can arrow functions have no parameters?
   A: Yes:

```
() => console.log("Hi");
```

629. **Q:** What if only one parameter?
   A: Parentheses can be omitted:

```
x => x * 2;
```

630. **Q:** Do arrow functions have a this?
   A: No, they inherit this from their lexical scope.
631. **Q:** Can arrow functions be constructors?
   A: No, they can’t be used with new.
632. **Q:** Can arrow functions have a block body?
   A: Yes, use {} and return inside.
633. **Q:** Can arrow functions return objects directly?
   A: Yes, wrap the object in parentheses:

```
() => ({name: "JS"})
```

634. **Q:** Can arrow functions access arguments?
   A: No.
635. **Q:** When should you not use arrow functions?
   A: When using this, arguments, or new.
636. **Q:** Do arrow functions have their own super?
   A: No.
637. **Q:** Can arrow functions be async?
   A: Yes:

```
const f = async () => {}
```

638. **Q:** Are arrow functions more efficient?
   A: Not necessarily; depends on use case.

639. **Q:** Can arrow functions be recursive?
   A: Yes, but must be assigned to a variable.
640. **Q:** Can you use return without {} in arrow functions?
   A: Yes:

```
x => x + 1
```

641. **Q:** Do arrow functions bind this lexically?
   A: Yes.
642. **Q:** Do they have a prototype?
   A: No.
643. **Q:** Are arrow functions hoisted?
   A: No.
644. **Q:** Can arrow functions be used in event handlers?
   A: Yes, but this won't refer to the element.
645. **Q:** Can arrow functions have multiple statements?
   A: Yes, use {}.
646. **Q:** Do arrow functions support await?
   A: Yes, if marked async.
647. **Q:** What is the difference between arrow function and regular function regarding this?
   A: Regular functions have their own this; arrows do not.
648. **Q:** Do arrow functions support method shorthand?
   A: No.
649. **Q:** Is this valid? const x = () => {};
   A: Yes.
650. **Q:** Is this valid? const x = (a, b) => return a + b;
   A: No, return can't be used like that without {}.

651. **Q:** What is scope in JavaScript?
A: Scope defines the visibility of variables.
652. **Q:** What are the types of scope?
A: Global scope, function scope, and block scope.
653. **Q:** What is function scope?
A: Variables declared inside a function are accessible only within that function.
654. **Q:** What is block scope?
A: Variables declared with let or const inside {} are accessible only inside that block.
655. **Q:** Does var have block scope?
A: No, it is function-scoped.
656. **Q:** What is a closure?
A: A function that remembers its outer scope, even after the outer function has finished.
657. **Q:** Why are closures useful?
A: For data privacy and maintaining state between function calls.
658. **Q:** Example of closure:
A:

```
function outer() {
 let count = 0;
 return function() { count++; return count; }
}
const counter = outer();
```

659. **Q:** What is an IIFE?
A: Immediately Invoked Function Expression.

660. **Q:** Syntax of IIFE?
    A:

```
(function() { ... })();
```

661. **Q:** Why use IIFE?
    A: To create a private scope and avoid polluting global scope.
662. **Q:** Can IIFE be arrow functions?
    A: Yes:

```
(() => { ... })();
```

663. **Q:** Can IIFE return values?
    A: Yes, it returns immediately.
664. **Q:** What is the arguments object?
    A: An array-like object containing all arguments passed to a function.
665. **Q:** Do arrow functions have arguments?
    A: No.
666. **Q:** How to access extra arguments without declaring parameters?
    A: Using arguments object (non-arrow functions only).
667. **Q:** Can functions have variable number of arguments?
    A: Yes, use rest parameters or arguments.
668. **Q:** What is a rest parameter?
    A:

```
function sum(...args) {}
```

669. **Q:** Can you use rest and normal parameters together?
    A: Yes, rest must be last.
670. **Q:** What is the difference between arguments and rest parameters?

A: arguments is not a real array and only in regular functions; rest is a real array.

671. **Q:** What is a pure function?
    A: A function that returns same output for same input and has no side effects.
672. **Q:** What is a side effect?
    A: Changing external state (e.g., modifying global variable, DOM).
673. **Q:** Are all functions pure?
    A: No, many perform side effects.
674. **Q:** What is currying?
    A: Transforming a function with multiple arguments into a series of unary functions.
675. **Q:** Example of currying:
    A:

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

Q: What is an async function?
A: A function declared with async that returns a promise.

676. **Q:** Syntax of async function?
    A:

```
async function getData() { ... }
```

678. **Q:** What does await do?
    A: Pauses the execution until a promise is resolved.

679. **Q:** Can await be used outside async?
A: No, it must be inside an async function (or top-level in modules).

680. **Q:** What happens if an async function throws an error?
A: It returns a rejected promise.

681. **Q:** How to handle errors in async/await?
A: Use try...catch.

682. **Q:** Example using async/await:
A:

```
async function fetchData() {
 try {
  let res = await fetch(url);
  let data = await res.json();
 } catch (err) {
  console.log(err);
 }
}
```

683. **Q:** Can arrow functions be async?
A: Yes:

```
const load = async () => {}
```

684. **Q:** What is the return value of async function?
A: A promise.

685. **Q:** What is the advantage of async/await over promises?
A: Cleaner syntax and easier to read.

686. **Q:** Can an async function call another async function?
A: Yes.

687. **Q:** What is function composition?
A: Combining multiple functions into one.

688. **Q:** Example of function composition?
A:

```
const compose = (f, g) => x => f(g(x));
```

689. **Q:** What is a generator function?
A: A function that can pause and resume using yield.

690. **Q:** Syntax of generator function?
A:

```
function* gen() { yield 1; yield 2; }
```

691. **Q:** How to call a generator function?
A: Use .next() on the returned generator object.

692. **Q:** What is .next()?
A: It resumes generator execution and returns { value, done }.

693. **Q:** Can you use yield in normal functions?
A: No, only in generator functions.

694. **Q:** What is tail call optimization?
A: A way to optimize recursive functions by reusing stack frames.

695. **Q:** Is tail call optimization supported in JS?
A: Partially, not widely supported.

696. **Q:** What is Function constructor?
A: Creates a new function dynamically:

```
const f = new Function("a", "b", "return a + b");
```

697. **Q:** Is it good to use new Function()?
A: Not recommended; it's like eval, and may have security issues.

698. **Q:** Can functions be stored in arrays?
    A: Yes.
699. **Q:** Can functions have properties?
    A: Yes, functions are objects.
700. **Q:** Can functions be used as methods?
    A: Yes, when defined inside an object.
701. **Q:** What is an object in JavaScript?
    A: A collection of key-value pairs.
702. **Q:** How do you create an object?
    A:

```
const obj = { name: "JS", age: 25 };
```

703. **Q:** How to access object properties?
    A: Using dot (obj.name) or bracket (obj["name"]) notation.
704. **Q:** How to add a new property to an object?
    A:

```
obj.newProp = "value";
```

705. **Q:** How to remove a property from an object?
    A: delete obj.prop;
706. **Q:** What is the in operator?
    A: Checks if a property exists: "name" in obj
707. **Q:** How to loop through object properties?
    A: Using for...in loop.
708. **Q:** What is Object.keys()?
    A: Returns an array of an object's keys.
709. **Q:** What is Object.values()?
    A: Returns an array of an object's values.

710. **Q:** What is Object.entries()?
 A: Returns an array of key-value pairs.

711. **Q:** What is a method?
 A: A function stored as a property in an object.

712. **Q:** Example of a method:
 A:

```
const obj = {
  greet() { return "Hello"; }
};
```

713. **Q:** What is this inside a method?
 A: Refers to the object that called the method.

714. **Q:** Can you nest objects?
 A: Yes, objects can contain other objects.

715. **Q:** What is object destructuring?
 A:

```
const { name } = obj;
```

716. **Q:** What is optional chaining (?.)?
 A: Safely accesses deeply nested properties.

717. **Q:** What is the Object.assign() method?
 A: Copies properties from source to target object.

718. **Q:** What is object spread syntax?
 A:

```
const newObj = { ...obj };
```

719. **Q:** Are object keys always strings?
 A: Yes, keys are strings or symbols.

720. **Q:** How to check if an object is empty?
 A:

Object.keys(obj).length === 0

721. **Q:** Can object values be functions?
   A: Yes.
722. **Q:** Can you compare objects using == or ===?
   A: No, it compares references, not content.
723. **Q:** How to clone an object?
   A: Object.assign({}, obj) or { ...obj }
724. **Q:** What is shallow copy?
   A: A copy where nested objects still share the same reference.
725. **Q:** What is deep copy?
   A: A copy where all nested objects are also cloned.
726. **Q:** What is prototype in JS?
   A: An object from which other objects inherit properties.
727. **Q:** How to access prototype of an object?
   A: Using __proto__ or Object.getPrototypeOf(obj)
728. **Q:** What is prototypal inheritance?
   A: Objects inherit properties from other objects.
729. **Q:** How to set prototype of an object?
   A: Using Object.setPrototypeOf()
730. **Q:** What is the prototype chain?
   A: A series of linked prototypes ending in null.
731. **Q:** What is hasOwnProperty()?
   A: Checks if a property exists directly on an object.
732. **Q:** What is a constructor function?
   A:

```
function Person(name) {
  this.name = name;
```

}

733. **Q:** How to create an object using constructor?
   A: const p = new Person("John");
734. **Q:** What is new keyword?
   A: It creates a new object and sets its prototype.
735. **Q:** How to add methods to constructor prototype?
   A:

Person.prototype.sayHi = function() {};

736. **Q:** Difference between prototype and __proto__?
   A: prototype is used on constructor functions; __proto__ is internal reference to an object's prototype.
737. **Q:** What is instanceof operator?
   A: Checks if an object is an instance of a constructor.
738. **Q:** Example of instanceof:
   A: john instanceof Person // true
739. **Q:** What is class in JS?
   A: A syntactic sugar over prototype-based inheritance.
740. **Q:** How to declare a class?
   A:

class Animal {}

741. **Q:** What is the constructor() method in classes?
   A: It initializes an object created from the class.
742. **Q:** How to create an object from a class?
   A: new Animal()
743. **Q:** Can classes have methods?
   A: Yes, declared inside the class.

744. **Q:** Can classes have static methods?
  A: Yes, using static keyword.

745. **Q:** What is super keyword?
  A: Calls the constructor of the parent class.

746. **Q:** How to create a class that extends another?
  A:

```
class Dog extends Animal {}
```

747. **Q:** Can constructors be overloaded in JS?
  A: No, only one constructor per class.

748. **Q:** What is class inheritance?
  A: One class inherits properties and methods from another.

749. **Q:** What is encapsulation in OOP?
  A: Bundling data and methods into a single unit (class).

750. **Q:** What is abstraction in OOP?
  A: Hiding complex details and showing only essential features.

751. **Q:** What are getters and setters?
  A: Special methods to get and set property values.

752. **Q:** How to define a getter?
  A:

```
get fullName() { return this.first + " " + this.last; }
```

753. **Q:** How to define a setter?
  A:

```
set fullName(name) { [this.first, this.last] = name.split(" "); }
```

754. **Q:** What are private class fields?
  A: Fields prefixed with # that can’t be accessed outside the class.

755. **Q:** Example of private field:
A:

```
class Person { #name = "John"; }
```

756. **Q:** What is Object.freeze()?
A: Prevents adding/removing/updating properties of an object.

757. **Q:** What is Object.seal()?
A: Prevents adding/removing but allows updating properties.

758. **Q:** What is Object.create()?
A: Creates a new object with the specified prototype.

759. **Q:** What is a mixin?
A: A way to add multiple behaviors to a class/object.

760. **Q:** How to check if a property exists?
A: Using "key" in obj or obj.hasOwnProperty("key")

761. **Q:** Can you define a method outside the class?
A: Yes, using prototype.

762. **Q:** What is Object.defineProperty()?
A: Defines a property with custom behavior (getters/setters, enumerable, etc.)

763. **Q:** What are property descriptors?
A: Settings like writable, configurable, enumerable.

764. **Q:** What is method chaining?
A: Calling multiple methods on the same object:

```
obj.method1().method2();
```

765. **Q:** What is object immutability?
A: When an object cannot be changed after creation.

766. **Q:** What is JSON.stringify()?
A: Converts an object to a JSON string.

767. **Q:** What is JSON.parse()?
A: Converts a JSON string to an object.

768. **Q:** Can object methods be serialized with JSON?
A: No, functions are not included in JSON.

769. **Q:** What is a factory function?
A: A function that returns an object:

```
function createUser(name) { return { name }; }
```

770. **Q:** Difference between class and factory function?
A: Classes use new and have prototypes; factories return plain objects.

771. **Q:** Can you have computed property names?
A: Yes:

```
const obj = { ["key" + 1]: "value" };
```

772. **Q:** Can you spread objects inside arrays?
A: Yes, using {...obj} inside [...].

773. **Q:** Can objects have Symbol keys?
A: Yes.

774. **Q:** What is object literal shorthand?
A: Omitting the key when key and variable name are same:

```
const name = "JS"; const obj = { name };
```

775. **Q:** What does Object.is() do?
A: Compares two values like ===, but treats NaN as equal to NaN.

776 . **Q:** What is DOM?
A: Document Object Model – a tree-like structure of the HTML document.

776. **Q:** What does document represent?
A: The entire HTML document as a JavaScript object.

777. **Q:** How to get an element by ID?
A: document.getElementById("id")

778. **Q:** How to get elements by class name?
A: document.getElementsByClassName("class")

779. **Q:** How to get elements by tag name?
A: document.getElementsByTagName("tag")

780. **Q:** What is querySelector()?
A: Returns the first element that matches a CSS selector.

781. **Q:** What is querySelectorAll()?
A: Returns all elements that match a CSS selector.

782. **Q:** What is innerHTML?
A: Gets/sets the HTML content of an element.

783. **Q:** What is textContent?
A: Gets/sets only the text of an element (no HTML).

784. **Q:** What is innerText?
A: Similar to textContent but considers styling.

785. **Q:** How to change element styles?
A:

element.style.color = "blue";

787. **Q:** How to change an attribute?
A: element.setAttribute("attr", "value")

788. **Q:** How to get an attribute?
 A: element.getAttribute("attr")
789. **Q:** How to remove an attribute?
 A: element.removeAttribute("attr")
790. **Q:** How to create a new element?
 A: document.createElement("tag")
791. **Q:** How to append an element to the DOM?
 A: parent.appendChild(child)
792. **Q:** How to remove an element?
 A: element.remove()
793. **Q:** How to replace an element?
 A: parent.replaceChild(newNode, oldNode)
794. **Q:** What is classList?
 A: Allows adding/removing/toggling CSS classes.
795. **Q:** How to add a class?
 A: element.classList.add("my-class")
796. **Q:** How to remove a class?
 A: element.classList.remove("my-class")
797. **Q:** How to toggle a class?
 A: element.classList.toggle("my-class")
798. **Q:** How to check if a class exists?
 A: element.classList.contains("my-class")
799. **Q:** What is parentNode?
 A: Returns the parent of the current node.
800. **Q:** What is children?
 A: Returns all child elements of a node.

851. **Q:** What is an event in JavaScript?
A: An action that occurs in the browser, like click, hover, or keypress.
852. **Q:** How to handle events in JavaScript?
A: Using event listeners or HTML event attributes.
853. **Q:** How to add an event listener?
A: element.addEventListener("click", function)
854. **Q:** How to remove an event listener?
A: element.removeEventListener("click", function)
855. **Q:** What is event object?
A: It contains information about the triggered event.
856. **Q:** What is event.target?
A: The element that triggered the event.
857. **Q:** What is event.currentTarget?
A: The element the event listener is attached to.
858. **Q:** How to prevent default action?
A: event.preventDefault()
859. **Q:** How to stop event bubbling?
A: event.stopPropagation()
860. **Q:** What is event bubbling?
A: Event propagates from child to parent elements.
861. **Q:** What is event capturing?
A: Event propagates from parent to child before the event target is reached.
862. **Q:** How to capture events in capturing phase?
A: Add third parameter true in addEventListener.
863. **Q:** What is once option in event listener?
A: Runs the handler only once.

864. **Q:** Example of once usage?
A:

```
element.addEventListener("click", fn, { once: true });
```

865. **Q:** What is keyboard event?
A: Events triggered by keyboard actions like keydown, keyup.

866. **Q:** What are mouse events?
A: Events like click, dblclick, mousedown, mousemove.

867. **Q:** What is mouseenter?
A: Triggers when mouse enters the element.

868. **Q:** What is mouseleave?
A: Triggers when mouse leaves the element.

869. **Q:** Difference between input and change event?
A: input triggers on each keystroke, change on losing focus.

870. **Q:** How to detect form submit?
A: form.addEventListener("submit", fn)

871. **Q:** What is preventDefault() in form submission?
A: Prevents form from submitting.

872. **Q:** What is event delegation?
A: Attaching a single event handler to a parent instead of multiple children.

873. **Q:** Why use event delegation?
A: It's efficient and handles dynamic elements.

874. **Q:** How to implement event delegation?
A: Use event.target to identify child elements inside a single listener.

875. **Q:** Can events be triggered manually?
A: Yes, using element.dispatchEvent(event)

876. **Q:** What is BOM?
 A: BOM allows JavaScript to interact with the browser window.
877. **Q:** What is window object?
 A: Global object representing the browser window.
878. **Q:** What is navigator object?
 A: Provides info about the browser.
879. **Q:** What is screen object?
 A: Gives info about the user’s screen (height, width, etc.)
880. **Q:** What is location object?
 A: Contains info about the current URL.
881. **Q:** How to reload a page?
 A: location.reload()
882. **Q:** How to redirect to another URL?
 A: location.href = "https://example.com"
883. **Q:** What is history object?
 A: Gives access to the browser’s session history.
884. **Q:** How to go back one page?
 A: history.back()
885. **Q:** How to go forward one page?
 A: history.forward()
886. **Q:** What is alert()?
 A: Shows a popup message.
887. **Q:** What is prompt()?
 A: Shows a dialog box for user input.
888. **Q:** What is confirm()?
 A: Shows a dialog with OK/Cancel and returns boolean.

889. **Q:** What is setTimeout()?
 A: Executes code after a delay.
890. **Q:** Example of setTimeout()
 A:

```
setTimeout(() => alert("Hello"), 1000);
```

891. **Q:** What is setInterval()?
 A: Runs code repeatedly at specified intervals.
892. **Q:** Example of setInterval()
 A:

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

893. **Q:** How to stop an interval?
 A: Using clearInterval(id)
894. **Q:** How to stop a timeout?
 A: Using clearTimeout(id)
895. **Q:** What is console.log()?
 A: Prints output to the browser console.
896. **Q:** What is console.error()?
 A: Logs an error message.
897. **Q:** What is console.table()?
 A: Logs data in tabular format.
898. **Q:** What is console.warn()?
 A: Logs a warning message.
899. **Q:** What is console.time() and console.timeEnd()?
 A: Measures execution time of code.
900. **Q:** How to open a new browser window?
 A: Using window.open("url")

901. **Q:** What is ES6?
A: ECMAScript 6 – major JS update with new syntax and features.

902. **Q:** What is let?
A: A block-scoped variable declaration.

903. **Q:** What is const?
A: Declares a block-scoped constant variable.

904. **Q:** Difference between var, let, and const?
A: var is function-scoped; let and const are block-scoped.

905. **Q:** What are template literals?
A: Strings using backticks with ${} for expressions.

906. **Q:** Example of template literal?
A: `` `Hello, ${name}` ``

907. **Q:** What is arrow function syntax?
A: Shorter function: const fn = () => {}

908. **Q:** Difference between regular and arrow functions?
A: Arrow functions don't have their own this.

909. **Q:** What are default parameters?
A: Parameters with default values.

910. **Q:** Example of default parameter?
A: function greet(name = "Guest")

911. **Q:** What is the spread operator?
A: Expands arrays or objects: ...arr

912. **Q:** What is the rest parameter?
A: Collects all remaining arguments: ...args

913. **Q:** What is object destructuring?
A: Extracts values from objects into variables.

914. **Q:** Example of object destructuring?
A: const {name, age} = user
915. **Q:** What is array destructuring?
A: Extracts elements into variables: [a, b] = arr
916. **Q:** What is a symbol?
A: A unique and immutable data type.
917. **Q:** What is Map in JS?
A: A collection of key-value pairs with any type of keys.
918. **Q:** What is Set in JS?
A: A collection of unique values.
919. **Q:** How to loop through a Map?
A: for (let [key, value] of map)
920. **Q:** What is a module?
A: A reusable JS file with exported functions/variables.
921. **Q:** How to export in a module?
A: export default or export { name }
922. **Q:** How to import from a module?
A: import name from './file.js'
923. **Q:** What is a class in JavaScript?
A: A blueprint for creating objects.
924. **Q:** How to define a class?
A:

```
class Person {
 constructor(name) {
  this.name = name;
 }
}
```

925. **Q:** What is inheritance in classes?
 A: One class extends another.

926. **Q:** Example of class inheritance?
 A:

```
class Student extends Person {}
```

927. **Q:** What is a constructor?
 A: A method called automatically when creating an object.

928. **Q:** What are static methods?
 A: Methods that belong to the class, not instance.

929. **Q:** What is super() in JS?
 A: Calls the constructor of the parent class.

930. **Q:** Can we have private properties in JS classes?
 A: Yes, using # before property name.

931. **Q:** What is optional chaining (?.)?
 A: Safely accesses nested properties.

932. **Q:** What is nullish coalescing (??)?
 A: Returns right-hand value if left is null or undefined.

933. **Q:** What is Object.entries()?
 A: Returns an array of key-value pairs.

934. **Q:** What is Object.values()?
 A: Returns array of values from an object.

935. **Q:** What is Object.keys()?
 A: Returns array of keys from an object.

936. **Q:** What is the includes() method?
 A: Checks if an array contains a value.

937. **Q:** What is the find() method?
 A: Returns the first element that satisfies a condition.

938. **Q:** What is the some() method?
A: Checks if at least one element matches a condition.

939. **Q:** What is the every() method?
A: Checks if all elements match a condition.

940. **Q:** What is the flat() method?
A: Flattens nested arrays into a single-level array.

941. **Q:** What is a generator function?
A: A function that can pause and resume.

942. **Q:** How to create a generator?
A: function* gen() { yield 1; }

943. **Q:** What is the yield keyword?
A: Pauses a generator function and returns value.

944. **Q:** What is a Proxy?
A: Allows customizing object operations (get, set, etc.)

945. **Q:** What is Reflect in JS?
A: Provides methods for interceptable JavaScript operations.

946. **Q:** What is tail call optimization?
A: Performance optimization for recursive functions.

947. **Q:** Does JavaScript support operator overloading?
A: No, not natively.

948. **Q:** What is BigInt?
A: A data type for very large integers.

949. **Q:** What is typeof BigInt(123)?
A: "bigint"

950. **Q:** Can we use _ in numeric literals?
A: Yes, for readability: 1_000_000

951. **Q:** What is a Promise?
A: An object representing future completion or failure.

952. **Q:** States of a Promise?
A: Pending, Fulfilled, Rejected.

953. **Q:** How to create a Promise?
A:

```
new Promise((resolve, reject) => { ... })
```

954. **Q:** What is then()?
A: Executes when the promise is fulfilled.

955. **Q:** What is catch()?
A: Handles errors or rejections.

956. **Q:** What is finally()?
A: Runs regardless of fulfillment or rejection.

957. **Q:** How to chain promises?
A: By returning another promise in then().

958. **Q:** What is Promise.all()?
A: Runs multiple promises in parallel and waits for all.

959. **Q:** What is Promise.race()?
A: Returns the first settled promise (fulfilled or rejected).

960. **Q:** What is Promise.any()?
A: Resolves as soon as one promise fulfills.

961. **Q:** What is Promise.allSettled()?
A: Waits for all promises to finish, regardless of result.

962. **Q:** What is async keyword?
A: Declares a function that returns a promise.

963. **Q:** What is await keyword?
A: Pauses async function until a promise resolves.

964. **Q:** Can await be used outside async functions?
 A: No, only inside async functions.
965. **Q:** Example of async/await?
 A:

```
async function getData() {
  const res = await fetch(url);
}
```

966. **Q:** Can you use try/catch with async/await?
 A: Yes, for error handling.
967. **Q:** What is the advantage of async/await?
 A: Cleaner, more readable promise-based code.
968. **Q:** Does await block other code?
 A: No, only pauses inside the async function.
969. **Q:** What is an API?
 A: Interface for applications to communicate.
970. **Q:** What is fetch() in JavaScript?
 A: Built-in method to make HTTP requests.
971. **Q:** Basic example of fetch()?
 A:

```
fetch(url).then(res => res.json()).then(data => ...)
```

972. **Q:** How to send POST data with fetch?
 A:

```
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(data)
})
```

973. **Q:** What is JSON?
    A: JavaScript Object Notation – lightweight data format.
974. **Q:** What is JSON.stringify()?
    A: Converts JS object to JSON string.
975. **Q:** What is JSON.parse()?
    A: Converts JSON string to JS object.
976. **Q:** What are HTTP methods?
    A: GET, POST, PUT, DELETE, etc.
977. **Q:** What are headers in fetch?
    A: Additional metadata sent with HTTP request.
978. **Q:** How to handle fetch errors?
    A: Using catch() or try/catch in async/await.
979. **Q:** What is status in fetch response?
    A: HTTP status code (e.g., 200, 404)
980. **Q:** What is CORS?
    A: Cross-Origin Resource Sharing – controls access to resources.
981. **Q:** Is JavaScript single-threaded?
    A: Yes, with a concurrency model based on event loop.
982. **Q:** What is the Event Loop?
    A: Handles async code by queuing and executing it later.
983. **Q:** What is the Call Stack?
    A: Stack of function calls to be executed.
984. **Q:** What is the Task Queue (Callback Queue)?
    A: Queue of async tasks waiting to run after the call stack clears.

985. **Q:** What is the Microtask Queue?
A: For promises and queueMicrotask(), runs before task queue.

986. **Q:** Order of execution: microtasks or tasks?
A: Microtasks are executed first.

987. **Q:** What is memory leak in JS?
A: Unused memory not released.

988. **Q:** How to prevent memory leaks?
A: Avoid global variables, detach event listeners, clear timers.

989. **Q:** What are JS design patterns?
A: Reusable solutions like Module, Singleton, Observer, etc.

990. **Q:** What is a polyfill?
A: Code that implements modern features in older browsers.

991. **Q:** What is transpilation?
A: Converting newer JS to older using tools like Babel.

992. **Q:** What is Babel?
A: A JS compiler to convert ES6+ to ES5.

993. **Q:** What is Webpack?
A: Module bundler for JavaScript.

994. **Q:** What is tree shaking?
A: Removing unused code in bundling process.

995. **Q:** What is minification?
A: Compressing code by removing spaces/comments.

996. **Q:** What is hoisting of functions?
A: Function declarations are hoisted entirely.

997. **Q:** Can JS be used for backend?
A: Yes, with Node.js.

998. **Q:** What is Node.js?

   A: JS runtime built on Chrome's V8 engine.

999. **Q:** What is NPM?

   A: Node Package Manager for JS packages.

1000. **Q:** Is JavaScript the same as Java?

   A: No, they are completely different languages.