# JAVASCRIPT

## Beginner To Advanced

## Day 5: Functions and Scope

- Function Declarations: Defining functions using the **function** keyword.
- Function Expressions: Defining functions as expressions, often assigned to variables.
- Arrow Functions: A more concise syntax for writing functions.
- Scope: The context in which variables are declared and can be accessed.
- Global Scope: Variables declared outside of any function, accessible throughout the entire program.
- Local Scope: Variables declared within a function, accessible only within that function.

## Day 7: Event Handling

- Events: Actions that occur as a result of user interactions or other triggers.
- Event Handlers: Functions that are executed when a specific event occurs.
- Event Listeners: Methods like **addEventListener()** to attach event handlers to elements.
- Event Object: An object containing information about the event, passed to event handler functions as an argument.
- Event Propagation: The order in which event handlers are executed, either capturing phase or bubbling phase.
- Event Delegation: Technique for handling events on multiple elements with a single event handler.

## Day 3: Loops and Iteration

- for Loop: Executes a block of code a specified number of times.
- While Loop: Executes a block of code while a specified condition is true.
- do...while Loop: Similar to the **while** loop, but the block of code is executed at least once before the condition is tested.
- for...in Loop: Iterates over the properties of an object.
- for...of Loop: Iterates over the values of an iterable object like arrays or strings.

## Day 4: Arrays and Objects

- Arrays: Ordered collections of values, accessed by numeric indices.
- Array Methods: Functions that can be called on arrays to manipulate their contents, such as **push(), pop(), shift(), unshift(), splice(), slice(),** etc.
- Objects: Collections of key-value pairs, where values can be accessed by keys.
- Object Methods: Functions that are properties of objects.

## Day 6: DOM Manipulation

- DOM (Document Object Model): A programming interface for web documents, representing the structure of an HTML document as a tree of objects.

- Selecting Elements: Methods like **getElementById(), getElementsByClassName(), getElementsByTagName(), querySelector(),** and **querySelectorAll()** to select elements from the DOM.

- Modifying Elements: Methods like **innerHTML, textContent, setAttribute(), classList,** etc., to modify the content and attributes of elements.

- Creating and Removing Elements: Methods like **createElement(), appendChild(), removeChild(),** etc., to dynamically create and remove elements from the DOM.

## Day 8: Asynchronous JavaScript

- Callbacks: Functions passed as arguments to other functions and executed later.
- Promises: Objects representing the eventual completion or failure of an asynchronous operation.
- async/await: Keywords used with asynchronous functions to write asynchronous code in a synchronous style.
- XHR (XMLHttpRequest): Object used to interact with servers and make HTTP requests from web browsers.
- Fetch API: Modern alternative to XHR for making HTTP requests in JavaScript.
- AJAX (Asynchronous JavaScript and XML): Technique for updating parts of a web page without reloading the whole page.

## Day 9: ES6 and Modern JavaScript

- ES6 (ECMAScript 2015): Major update to the JavaScript language, introducing new syntax and features.
- Arrow Functions: A more concise syntax for writing functions.
- Template Literals: Strings that allow embedded expressions.
- Destructuring Assignment: Extracting values from arrays or objects and assigning them to variables.
- Spread Operator: Expands an iterable (like an array) into individual elements.
- Classes & Inheritance: Prototypal inheritance in JavaScript using class syntax.
- Modules: Encapsulating code into reusable modules using import and export statements.

# Day 10: Advanced JavaScript Concepts

- Closures: Functions that remember the scope in which they were created, even after that scope has closed.
- Prototypes and Prototypal Inheritance: The mechanism by which JavaScript objects inherit features from one another.
- Context (this): A reference to the object that owns the currently executing code.
- Execution Context and Hoisting: The context in which JavaScript code is executed and the process of moving variable and function declarations to the top of their containing scope.
- Event Loop: The mechanism that allows JavaScript to perform non-blocking operations.
- Memory Management: How JavaScript manages memory allocation and deallocation, including garbage collection.

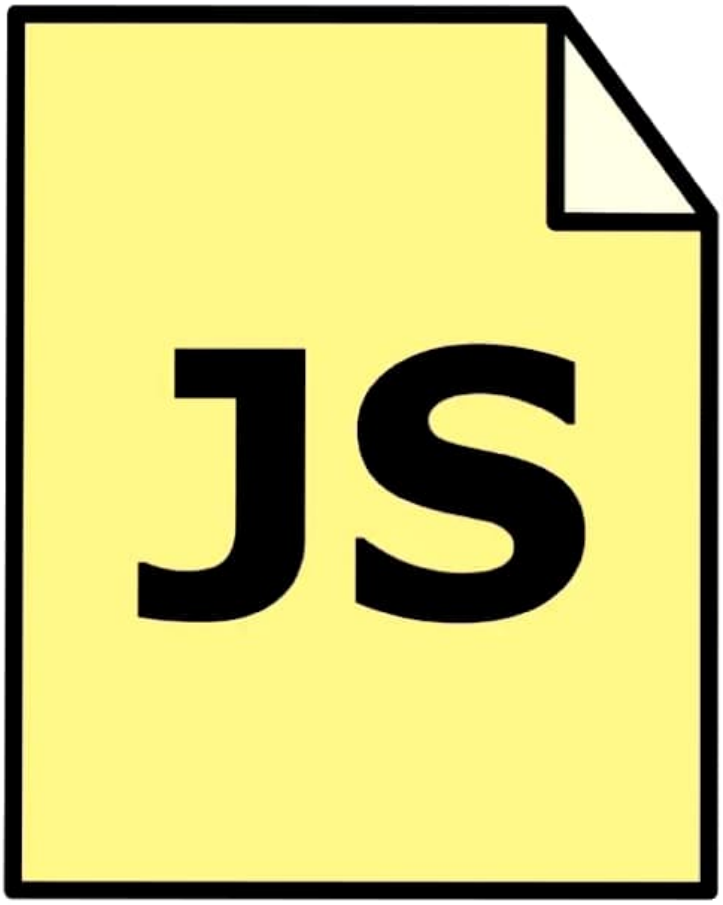
# Day 11 & 12: Functional Programming & Advanced JS

- Functional Programming: A programming paradigm focused on building software by composing pure functions and avoiding shared state, mutable data, and side-effects.
- Pure Functions: Functions that return the same output for the same input and do not produce side effects.
- Immutability: The principle that data should not be changed after it is created.
- Higher-Order Functions: Functions that take other functions as arguments or return functions.
- Map, Filter, & Reduce: Higher-order functions commonly used in functional programming for transforming and aggregating data.
- Recursion: A technique where a function calls itself in order to solve smaller instances of the same problem.
- Module Patterns: Techniques for encapsulating and organizing code into modules.
- Singleton Pattern: A design pattern that restricts the instantiation of a class to a single instance.
- Observer Pattern: A design pattern where an object, called the subject, maintains a list of its dependents, called observers, and notifies them of any state changes.
- Promises & Async/Await Patterns: Patterns for managing asynchronous code and handling asynchronous operations in JavaScript.
- Memoization: A technique of storing the results of expensive function calls and returning the cached result when the same inputs occur again.


WEB DEVELOPER

@COMPUTER__PROGRAMMER

# 50 JavaScript Interview Q/A




# @COMPUTER__PROGRAMMER

**1. What is JavaScript?**

- JavaScript is a high-level, interpreted programming language primarily used for building interactive web applications.

**2. What are the data types in JavaScript?**

- Primitive data types: number, string, boolean, null, undefined, symbol.
- Non-primitive data type: object.

3. **What is the difference between == and === operators in JavaScript?**

- == checks for equality after type coercion, while === checks for equality without type coercion.

4. **What is hoisting in JavaScript?**

- Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their containing scope during compilation.

5. **What is closure in JavaScript?**

- A closure is the combination of a function bundled together with references to its surrounding state, allowing it to retain access to variables from its lexical scope even when called outside that scope.

6. **Explain event delegation in JavaScript.**

- Event delegation is a technique of handling events at a higher level in the DOM tree and letting events propagate to their target element.

7. **What is a promise in JavaScript?**

- A promise represents the eventual completion or failure of an asynchronous operation, and its resulting value.

8. **What is the purpose of the this keyword in JavaScript?**

- **this** refers to the object to which a function or method belongs, or the object that is currently being operated on.

9. **What is a callback function in JavaScript?**

- A callback function is a function passed as an argument to another function, which is then invoked inside the outer function to complete some kind of routine or action.

19. **What is the difference between var, let, and const**

- **var** declares variables with function scope, **let** declares variables with block scope, and **const** declares variables that cannot be reassigned.

20. **What are some ways to handle asynchronous code in JavaScript**

- Callbacks, Promises, Async/Await.

21. **What is the event loop in JavaScript?**

- The event loop is a single-threaded mechanism that processes tasks in a queue, handling asynchronous operations like callbacks, promises, and I/O operations.

22. **Explain the difference between document.ready() and window.onload() in JavaScript.**

- **document.ready()** is an event that fires when the DOM is fully loaded, while **window.onload()** is an event that fires when all assets have loaded.

23. **What is a higher-order function in JavaScript?**

- A higher-order function is a function that takes one or more functions as arguments or returns a function as its result.

24. **What is the purpose of the map method in JavaScript?**

- The **map** method creates a new array with the results of calling a provided function on every element in the calling array.

25. **What is the difference between Object.keys() and Object.values()?**

- **Object.keys()** returns an array of a given object's own enumerable property names, while **Object.values()** returns an array of a given object's own enumerable property values.

26. **Explain the concept of destructuring in JavaScript.**

- Destructuring is a JavaScript expression that allows extracting data from arrays, objects, and function parameters and binding it to variables.

27. **What is the purpose of the find method in JavaScript?**

- The **find** method returns the first element in an array that satisfies the provided testing function.

**10. What is event bubbling in JavaScript?**

- Event bubbling is the process where an event propagates through the DOM tree from the innermost target element up to the document root.

11. **Explain the concept of prototypal inheritance in JavaScript.**

- Prototypal inheritance is a way of creating objects based on other objects, where objects inherit properties and methods directly from other objects.

12. **What is the purpose of the bind method in JavaScript?**

- The **bind** method creates a new function that, when called, has its **this** keyword set to a specific value.

13. **What is the difference between null and undefined in JavaScript?**

- **null** is an intentional absence of any value, while **undefined** means a variable has been declared but has not yet been assigned a value.

14. **What is the purpose of the let keyword in JavaScript?**

- The **let** keyword declares block-scoped variables, which means the variable is only accessible within the block it is defined in.

15. **What is the purpose of the const keyword in JavaScript?**

- The **const** keyword declares variables that cannot be re-assigned a new value once initialized.

16. **What is an arrow function in JavaScript?**

- An arrow function is a shorthand syntax for writing function expressions, providing a more concise way to define functions.

17. **Explain the concept of asynchronous programming in JavaScript.**

- Asynchronous programming allows code to run non-sequentially, enabling tasks to be executed concurrently and improving performance by not blocking the main execution thread.

18. **What is the forEach method in JavaScript?**

- The **forEach** method executes a provided function once for each array element.

37. **What is the purpose of the Symbol data type in JavaScript?**

- The **Symbol** data type represents a unique and immutable value that may be used as the key of an Object property.

38. **What is the arguments object in JavaScript?**

- The **arguments** object is an array-like object accessible inside functions that contains the values of the arguments passed to that function

39. **What is the purpose of the String.prototype.split() method in JavaScript?**

- The **split()** method splits a string object into an array of strings by separating the string into substrings.

40. **What is the purpose of the String.prototype.trim() method in JavaScript?**

- The **trim()** method removes whitespace from both ends of a string.

41. **What is the purpose of the String.prototype.replace() method in JavaScript?**

- The **replace()** method returns a new string with some or all matches of a pattern replaced by a replacement.

42. **What is the purpose of the String.prototype.charAt() method in JavaScript?**

- The **charAt()** method returns the character at a specified index in a string.

43. **What is the purpose of the String.prototype.concat() method in JavaScript?**

- The **concat()** method concatenates the string arguments to the calling string and returns a new string.

44. **What is the purpose of the String.prototype.indexOf() method in JavaScript?**

- The **indexOf()** method returns the position of the first occurrence of a specified value in a string.

45. **What is the purpose of the String.prototype.toUpperCase() method in JavaScript?**

- The **toUpperCase()** method returns the calling string value converted to uppercase.

**28. What is the difference between == and === operators in JavaScript**

- == performs type coercion before checking equality, while === strictly checks for equality without type conversion.

29. **What is the purpose of the reduce method in JavaScript?**

- The **reduce** method applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value.

30. **What is the purpose of the filter method in JavaScript?**

- The **filter** method creates a new array with all elements that pass the test implemented by the provided function.

31. **What is the purpose of the some method in JavaScript?**

- The **some** method tests whether at least one element in the array passes the test implemented by the provided function.

32. **What is the purpose of the every method in JavaScript?**

- The **every** method tests whether all elements in the array pass the test implemented by the provided function.

33. **What is the difference between setTimeout and setInterval functions in JavaScript?**

- **setTimeout** executes a function once after a specified time interval, while **setInterval** repeatedly executes a function at specified time intervals.

34. **What is the purpose of the slice method in JavaScript?**

- The **slice** method returns a shallow copy of a portion of an array into a new array object selected from start to end (end not included).

35. **What is the purpose of the splice method in JavaScript?**

- The **splice** method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

36. **What is a generator function in JavaScript?**

- A generator function is a special type of function that can pause and resume its execution, allowing iterative algorithms to be written in a more intuitive way.

**46. What is the purpose of the String.prototype.toLowerCase() method in JavaScript?**

- The **toLowerCase()** method returns the calling string value converted to lowercase.

47. **What is the purpose of the String.prototype.slice() method in JavaScript?**

- The **slice()** method extracts a section of a string and returns it as a new string.

48. **What is the purpose of the String.prototype.substr() method in JavaScript?**

- The **substr()** method returns the characters in a string beginning at the specified location through the specified number of characters.

49. **What is the purpose of the String.prototype.startsWith() method in JavaScript?**

- The **startsWith()** method determines whether a string begins with the characters of a specified string, returning true or false as appropriate.

50. **What is the purpose of the String.prototype.endsWith() method in JavaScript?**

- The **endsWith()** method determines whether a string ends with the characters of a specified string, returning true or false as appropriate.

# All JavaScript Methods

@coding__comics

# String Methods

- charAt(index)

```
let str = "Hello";
console.log(str.charAt(1)); // "e"
```

- includes(searchString)

```
let str = "Hello World";
console.log(str.includes("World")); // true
```

- replace(searchValue, newValue)

```
let str = "Hello World";
console.log(str.replace("World", "JavaScript")); // "Hello JavaScript"
```

- slice(start, end)

```
let str = "Hello World";
console.log(str.slice(0, 5)); // "Hello"
```

- toUpperCase()

```
let str = "hello";
console.log(str.toUpperCase()); // "HELLO"
```

- split(separator)

```
let str = "a,b,c,d";
console.log(str.split(",")); // ["a", "b", "c", "d"]
```

# Array Methods

- concat(array1, array2)

```
let arr1 = [1, 2];
let arr2 = [3, 4];
console.log(arr1.concat(arr2)); // [1, 2, 3, 4]
```

- filter(callback)

```
let arr = [1, 2, 3, 4];
console.log(arr.filter(num => num > 2)); // [3, 4]
```

- find(callback)

```
let arr = [1, 2, 3, 4];
console.log(arr.find(num => num > 2)); // 3
```

- map(callback)

```
let arr = [1, 2, 3, 4];
console.log(arr.map(num => num * 2)); // [2, 4, 6, 8]
```

- **push(element)**

```
let arr = [1, 2];
arr.push(3);
console.log(arr); // [1, 2, 3]
```

- **pop()**

```
let arr = [1, 2, 3];
arr.pop();
console.log(arr); // [1, 2]
```

## Object Methods

- **assign(target, ...sources)**

```
let target = { a: 1 };
let source = { b: 2 };
Object.assign(target, source);
console.log(target); // { a: 1, b: 2 }
```

- **keys(obj)**

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

- values(obj)

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

- entries(obj)

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

# Math Methods

- abs(x)

```
console.log(Math.abs(-5)); // 5
```

- ceil(x)

```
console.log(Math.ceil(4.2)); // 5
```

- floor(x)

```
console.log(Math.floor(4.8)); // 4
```

- max(x1, x2, ...)

```
console.log(Math.max(1, 2, 3)); // 3
```

- random()

```
console.log(Math.random()); // Random number between 0 and 1
```

# Date Methods

- getDate()

```
let date = new Date();
console.log(date.getDate()); // Current day of the month (1-31)
```

- getFullYear()

```
let date = new Date();
console.log(date.getFullYear()); // Current year (e.g., 2024)
```

- getTime()

```
let date = new Date();
console.log(date.getTime()); // Milliseconds since January 1, 1970
```

- setDate(day)

```
let date = new Date();
date.setDate(15);
console.log(date); // Date set to the 15th of the current month
```

# JSON Methods

- parse(text)

```
let jsonString = '{"name": "John", "age": 30}';
let obj = JSON.parse(jsonString);
console.log(obj); // { name: "John", age: 30 }
```

- stringify(value)

```
let obj = { name: "John", age: 30 };
let jsonString = JSON.stringify(obj);
console.log(jsonString); // '{"name":"John","age":30}'
```

# Promise Methods

- all(promises)

```
let p1 = Promise.resolve(1);
let p2 = Promise.resolve(2);
Promise.all([p1, p2]).then(result => console.log(result)); // [1, 2]
```

- race(promises)

```
let p1 = new Promise((resolve) => setTimeout(resolve, 500, "One"));
let p2 = new Promise((resolve) => setTimeout(resolve, 100, "Two"));
Promise.race([p1, p2]).then(result => console.log(result)); // "Two"
```

---

“ Code is like humor. When you have to explain it, it’s bad ”

Coding Comics


WEB DEVELOPER

@COMPUTER__PROGRAMMER

# JAVASCRIPT CHEATSHEET 2024

## 3. OPERATORS

```
let sum = 5 + 10; // Arithmetic
let isEqual = 5 == '5'; // Equality
let isStrictEqual = 5 === '5'; // Strict Equality
let andOperator = true && false; // Logical AND
let orOperator = true || false; // Logical OR
```

## 4. CONDITIONAL STATEMENTS

```
if (x > 10) {
  console.log("x is greater than 10");
} else if (x === 10) {
  console.log("x is 10");
} else {
  console.log("x is less than 10");
}
```

## 5. SWITCH STATEMENT

```
switch (day) {
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  default:
    console.log("Another day");
}
```

## 7. FUNCTIONS

```
function greet(name) {
  return `Hello, ${name}`;
}

// Arrow function
const greetArrow = (name) => `Hello, ${name}`;
```

## 8. OBJECTS

```
let person = {
  name: "Alice",
  age: 30,
  greet: function() {
    console.log("Hello");
  }
};
person.greet(); // Access method
console.log(person.name); // Access property
```

# 6. LOOPS

```
// For loop
for (let i = 0; i < 5; i++) {
  console.log(i);
}

// While loop
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

## 9. ARRAYS

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // Access element

fruits.push("orange"); // Add element
fruits.pop(); // Remove element
```

## 10. ARRAY METHODS

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

// Map
let squared = numbers.map(num => num * num);

// Filter
let evens = numbers.filter(num => num % 2 === 0);

// Reduce
let sum = numbers.reduce((total, num) => total + num, 0);
```