**The JavaScript Mystery**

**Introduction to JavaScript:**

* ­­­What is JavaScript?

Ans: JavaScript is a high level, interpreated,mother language of web…

JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.

ECMA-262 is the official name of the standard. ECMAScript is the official name of the language.

* First js was called “Toy Language before 2009”.
* JS can change html element.
* JS can change html attribute value.
* JavaScript Can Change HTML Styles (CSS).
* The <script> tag in html is usually placed in the end of the body. It can be placed in the <head> tag as well. Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display. Suppose there is a function to add 1 to 10000000 numbers. Now if the code runs in the browser than it will take few times to add those huge numbers. In that time JS will hold all the code rendering which is below of head tag. After adding all those numbers than the other code will run and show in the UI. That might not be user friendly. But if we place the <script> tag in the end of <body> tag, than the top other codes or section will show in the UI first and after rendering all the designs in the UI, it will run the JS code.
* Js code can be written in the html file inside the <script> tag. And it can also be used by the external js file. We have to import it. It will work same. But defining the js file externally can be more beneficial in many ways:

1. It separates HTML and code
2. It makes HTML and JavaScript easier to read and maintain
3. Cached JavaScript files can speed up page loads. It means if a user first visit any website then all the js files are saved in the cache memory and if the user visit that website again than it doesn’t take much time to load.
4. Code Reusability
5. Ease of Maintenance: External JavaScript files make it easier to manage and maintain code. Instead of searching through HTML files for JavaScript code, developers can focus on the standalone JavaScript files, improving readability and organization.
6. Collaboration: When working in a team, using external JavaScript files allows developers to work on different parts of a project simultaneously without interfering with each other's code. Version control systems like Git can also be more effectively utilized.
7. Browser Compatibility: External JavaScript files can be linked using standard HTML tags, ensuring better compatibility across different browsers and devices. This helps avoid potential inconsistencies and bugs that might arise from inline JavaScript code.

**JavaScript Functions and Events:**

A JavaScript function is a block of JavaScript code, that can be executed when "called" for. For example, a function can be called when an event occurs, like when the user clicks a button.

**Operators in JavaScript:**

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Unary Operators
7. Ternary Operator (Conditional Operator)
8. String Operators
9. Comma Operator

**Comparison Operators:**

1. Greater than : >
2. Less than : <
3. Greater than or equal : >=
4. Less than or equal : <=
5. Equal : == or ===
6. Not equal : !=
7. And : &&
8. Or : ||

**Conditional Operators:**

1. if Statement
2. if...else Statement
3. else if Statement
4. Nested if Statements
5. Ternary Operator (Conditional Operator)
6. Switch Statement
7. Logical not

**If Statement:** The if statement executes a statement if a specified condition is truthy.

**Else statement:** If the condition is falsy then else condition will execute.

**JavaScript Array Methods:**

1. **concat:** Combines two or more arrays, returning a new array without modifying the original arrays. It does not change the existing arrays but instead returns a new array that contains elements from all the arrays provided.
2. **copyWithin:** Copies a sequence of elements within the array. It overwrites existing elements and shifts others to accommodate the copied elements. It modifies the array in place and returns a reference to the modified array.
3. **entries:** Returns an iterator object with key/value pairs for each index in the array. It allows iterating over the array's keys and values using a loop or other iterable methods.
4. **every:** Checks if all elements in the array pass a test provided by a function. It returns a boolean value indicating whether all elements satisfy the specified condition.
5. **fill:** Fills all elements of an array with a static value. It modifies the array in place and returns a reference to the modified array.
6. **filter:** Creates a new array with elements that pass a test provided by a function. It returns a new array containing only the elements for which the provided function returns true.
7. **find:** Returns the first element in an array that satisfies a provided condition. It returns the value of the first element in the array that satisfies the provided testing function, or undefined if no such element is found.
8. **findIndex:** Returns the index of the first element in the array that satisfies a provided condition. It returns the index of the first element in the array that satisfies the provided testing function, or -1 if no such element is found.
9. **flat:** Creates a new array with all sub-array elements concatenated into it recursively up to the specified depth. It flattens nested arrays into a single-level array.
10. **flatMap:** Maps each element using a function and flattens the result into a new array. It first maps each element using a mapping function, then flattens the result into a new array.
11. **forEach:** Executes a provided function once for each array element. It iterates over each element in the array and executes the provided function with each element as an argument.
12. **includes:** Determines whether an array contains a certain element. It returns true if the array contains the specified element, false otherwise.
13. **indexOf:** Returns the first index at which a given element can be found in the array, or -1 if it is not present. It searches the array from beginning to end and returns the index of the first occurrence of the specified element.
14. **join:** Joins all elements of an array into a string. It concatenates all elements of the array using a specified separator and returns the resulting string.
15. **keys:** Returns an iterator of the array's keys. It returns a new Array Iterator object that contains the keys for each index in the array.
16. **lastIndexOf:** Returns the last index at which a given element can be found in the array, or -1 if it is not present. It searches the array from end to beginning and returns the index of the last occurrence of the specified element.
17. **map:** Creates a new array with the results of calling a provided function on every element in the calling array. It returns a new array containing the results of applying the provided function to each element of the original array.
18. **pop:** Removes the last element from an array and returns that element. It modifies the array by removing the last element and returns the removed element.
19. **push:** Adds one or more elements to the end of an array and returns the new length of the array. It modifies the array by appending new elements to the end and returns the new length of the array.
20. **reduce:** Reduces the array to a single value by applying a function against an accumulator and each element in the array. It iterates over each element in the array, accumulating a single value based on the result of the provided function.
21. **reduceRight:** Reduces the array from right to left to a single value by applying a function against an accumulator and each element in the array. It iterates over each element in the array from right to left, accumulating a single value based on the result of the provided function.
22. **reverse:** Reverses the order of the elements in an array in place. It modifies the array by reversing the order of its elements and returns a reference to the modified array.
23. **shift:** Removes the first element from an array and returns that removed element. It modifies the array by removing the first element and shifting all other elements to a lower index.
24. **slice:** Extracts a section of an array and returns a new array. It returns a shallow copy of a portion of an array into a new array object selected from start to end.
25. **some:** Checks if at least one element in the array passes a test provided by a function. It returns true if at least one element satisfies the provided testing function, false otherwise.
26. **sort:** Sorts the elements of an array in place and returns the sorted array. It modifies the array by sorting its elements according to the provided sorting criteria and returns a reference to the sorted array.
27. **splice:** Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place. It modifies the array by removing or replacing existing elements and returns an array containing the removed elements.
28. **toLocaleString:** Returns a string representing the elements of the array. It returns a localized string representing the array and its elements using the system's default locale settings.
29. **toString:** Returns a string representing the specified array and its elements. It returns a string representing the array and its elements separated by commas.
30. **unshift:** Adds one or more elements to the beginning of an array and returns the new length of the array. It modifies the array by adding new elements to the beginning and returns the new length of the array.
31. **values:** Returns an iterator of the array's values. It returns a new Array Iterator object that contains the values for each index in the array.

**JavaScript Looping Methods:**

**for loop:** This is the most basic looping mechanism in JavaScript. It consists of three parts: initialization, condition, and iteration. The loop continues executing as long as the condition evaluates to true.

**for...in loop:** This loop iterates over the enumerable properties of an object, including inherited properties from its prototype chain. It should be used with caution with arrays because it iterates over all enumerable properties, not just array elements, and may not iterate in the expected order.

**for...of loop:** Introduced in ES6, this loop iterates over inerrable objects such as arrays, strings, maps, sets, etc. It provides a simpler syntax compared to traditional for loops.

**Array.forEach():** This method executes a provided function once for each array element. It is commonly used for side effects such as logging or updating external state.

**Array.map():** This method creates a new array with the results of calling a provided function on every element in the calling array. It is used when you need to transform each element of an array into something else.

**Array.filter():** This method creates a new array with all elements that pass the test implemented by the provided function. It is commonly used to filter out elements based on certain criteria.

**Array.reduce():** This method executes a reducer function on each element of the array, resulting in a single output value. It is often used to perform computations on an array and return a single value.

**Array.reduceRight():** This method is similar to Array.reduce(), but it processes the array from right to left instead of left to right.

**JavaScript String Methods:**

1. **charAt():** Returns the character at the specified index (position) in a string.
2. **charCodeAt():** Returns the Unicode value of the character at the specified index in a string.
3. **concat():** Concatenates one or more strings to the end of another string and returns a new string.
4. **includes():** Determines whether one string may be found within another string, returning true or false.
5. **indexOf():** Returns the index within the calling string object of the first occurrence of the specified value, starting the search at fromIndex.
6. **lastIndexOf():** Returns the index within the calling string object of the last occurrence of the specified value, or -1 if not found.
7. **match():** Retrieves the result of matching a string against a regular expression.
8. **repeat():** Returns a new string with a specified number of copies of the string it was called on.
9. **replace():** Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced.
10. **search():** Executes the search for a match between a regular expression and this String object.
11. **slice():** Extracts a section of a string and returns it as a new string, without modifying the original string.
12. **split():** Splits a string into an array of substrings based on a specified separator.
13. **startsWith():** Determines whether a string begins with the characters of another string, returning true or false.
14. **substr():** Returns the characters in a string beginning at the specified location through the specified number of characters.
15. **substring():** Returns the characters in a string between two indexes into the string.
16. **toLocaleLowerCase():** Returns the calling string value converted to lower case, according to any locale-specific case mappings.
17. **toLocaleUpperCase():** Returns the calling string value converted to upper case, according to any locale-specific case mappings.
18. **toLowerCase():** Returns the calling string value converted to lowercase.
19. **toString():** Returns the value of a String object.
20. **toUpperCase():** Returns the calling string value converted to uppercase. trim(): Removes whitespace from both ends of a string.
21. **valueOf():** Returns the primitive value of a String object.
22. **join():** Joins all elements of an array into a string, separated by a specified separator.

**JavaScript Objects:**

An object in JavaScript is a data structure that allows you to store collections of key-value pairs. Each key is a string (or a symbol), and each value can be any data type, including other objects, arrays, functions, etc.

**Object Creation:**

**Object Literal:**

* Objects can be created using object literal notation {}.
* Key-value pairs are defined within curly braces {}.
* Example: { key1: value1, key2: value2 }.

**Object Constructor:**

* Objects can also be created using the new Object() constructor.
* Properties can be added individually after object creation.
* Example: const obj = new Object(); obj.key = value;.

**Accessing Properties:**

1. **Dot Notation:**

* Use dot notation (obj.key) to access properties with simple names that are valid identifiers.

1. **Bracket Notation:**

* Use bracket notation (obj['key']) to access properties with dynamic or special characters.

**Property Manipulation:**

**Adding or Modifying Properties:** Properties can be added or modified on an object after creation by assigning values to them.

**Deleting Properties:** Properties can be removed from an object using the delete operator.

**Object Methods:**

1. **Object.keys():** Returns an array containing the names of all enumerable properties of an object.
2. **Object.values():** Returns an array containing the values of all enumerable properties of an object.
3. **Object.entries():** Returns an array containing all enumerable property [key, value] pairs of an object.
4. **Object.assign():** Copies the values of all enumerable own properties from one or more source objects to a target object.

**Function in JavaScrict:**

What is function?

Ans: Function is a block of code, a set of statements that performs a task when it is called and it maintain a relationship with input and output.