**switch statement -**

* In JavaScript, the switch statement is a control flow statement that allows you to execute different blocks of code based on the value of a variable or expression.
* It provides an alternative to using a long chain of if-else if-else statements, making your code more readable and often more efficient when you have many possible conditions to check against the same variable.
* Syntax -

**switch (expression) {**

**case value1:**

**// Code to be executed if expression === value1**

**break;**

**case value2:**

**// Code to be executed if expression === value2**

**break;**

**// ... more cases**

**default:**

**// Code to be executed if none of the cases match**

**}**

**switch -**

* This is the value that the switch statement evaluates.
* It can be any JavaScript expression that results in a primitive value (number, string, boolean, null, undefined) or an object (though typically primitive values are used).
* The switch statement performs a strict equality comparison (===) between the expression and each case value.

**break -**

* The break keyword is crucial within a switch statement.
* Once a matching case is found and its code is executed, break immediately terminates the switch statement.

**default -**

* The default keyword is optional.
* If no case matches the expression, the code block within the default case is executed.
* It's similar to the else block in an if-else statement.

**//Create one calculator to perform arithmetic operations - +, -, \* , /**

**let num1 = 10**

**let num2 = 5**

**let operator = '/'**

**switch(operator){**

**case '+' : console.log(`Add - ${num1} + ${num2} =  ${num1 + num2}`)**

**break;**

**case '-' : console.log(`Subtract - ${num1} - ${num2} = ${num1 - num2}`)**

**break;**

**case '\*' : console.log(`Multiplication : ${num1} \* ${num2} = ${num1 \* num2}`)**

**break;**

**case '/' : console.log(`Division : ${num1} / ${num2} = ${num1 / num2}`)**

**break;**

**default : console.log(`Select valid operator +,-,\*,/ - You entered ${operator}`)**

**}**

**//Grading System using switch statement**

**Condition:**

**If the score is 90 or higher, the grade is "A".**

**Else if the score is 80 or higher, the grade is "B".**

**Else if the score is 70 or higher, the grade is "C".**

**Else if the score is 60 or higher, the grade is "D".**

**Otherwise (if below 60), the grade is "F - You need to study more!".**

**Array In JavaScript**

* An Array is a data structure in JavaScript that allows you to store multiple values in a single variable.
* Array can store different type of value - number, strings, booleans, objects or even array
* Array use square bracket to store multiple values - [ ]

Ex.

const fruits = [‘apple’, ‘grapes’, ‘orange’, ‘cherry’]

const numbers = [1,2,3,4,5,6,7,8,9,10]

* We can access each value of the array using index number (position), and which is start from 0
* Why use Arrays -

To store lists of data

To group related values together

**Array Methods -**

1. **Length property** - Return the total number of item

Syntax - arrayName.length

1. **push ()** - This method is used to add one or more element at the end of an array

Syntax - arrayName.push(ele1, ele2……eleN)

1. **pop ()** -This method removes the last element from an array and returns it.

Syntax - arrayName.pop()

1. **unshift ()-**Inserts new elements at the start of an array, and returns the new length of the array.

Syntax - arrayName.unsshift(ele1, ele2……eleN)

1. **shift ()-**This method removes first element (0th index) from an array

Syntax - arrayName.shift()

1. **reverse () –**Reverses the elements in an array in place. This method mutates the array and returns a reference to the same array.

Syntax - arrayName.reverse()

1. **join () -**This method is used to join all array elements in a single string value. This method return a string not array. In join() method we use separators

Syntax - arrayName.join(separator)

1. **flat () -**This method used to flatten the nested array (reduce the nesting level of array)

Syntax - arrayName.flat(depth)

* depth – An integer value to specify how deep a nested array structured should be flatten
* Infinity

1. **Splice()** – This method is used to add or remove array element on the specified index number.

This method modify the existing array

Syntax - arrayName.splice()

To remove element –

arrayName.splice(startIndexNumber,deleteCount)

* + - The zero-based location in the array from which to start removing elements.
    - The number of elements to remove. (negative deleteCount not allowed)

To add element –

arrayName.splice(startIndexNumber,deleteCount, newElement)

* + - The zero-based location in the array from which to start removing elements.
    - The number of elements to remove. (negative deleteCount not allowed)
    - Mentioned the new element

To add and remove -

arrayName.splice(startIndexNumber, deleteCount, newElement)

* + - The zero-based location in the array from which to start removing elements.
    - The number of elements to remove. (negative deleteCount not allowed) Mentioned the new element

1. **slice() -** Returns a portion of an array without modifying the original array.

Syntax:

array.slice(start, end)

* + start → index to begin (inclusive).
  + end → index to stop (exclusive).
  + If end is omitted, it goes till the end.

1. **Sort() method –** The sort() method in JavaScript is used to **arrange the elements of an array in order**.

It **modifies the original array** and returns the **sorted array**.

Syntax:

array.sort(start, end)

1. **map () Method –** The map() method creates a new array populated with the results of calling a provided function on every element in the calling array. It does not modify the original array.

Key Characteristics:

* + - Returns a new array: The most important point is that map() always produces a brand new array.
    - Transformation: It's used when you want to transform each element of an array into something else.
    - Same length: The new array created by map() will always have the same length as the original array.

Syntax:

array.map(callback function)

1. **filter () method –**  The filter() method creates a new array with all elements that pass the test implemented by the provided function. It does not modify the original array.

● Key Characteristics:

* + - Returns a new array: Like map(), filter() always returns a new array.
    - Conditional selection: It's used when you want to select a subset of elements from an array based on a condition.
    - Equal or shorter length: The new array created by filter() will have a length equal to or shorter than the original array.

Syntax:

Array.filter(callback function)