Template literals

Template literals allow you to embed expressions and variables directly within the string using **${}**. This makes **string interpolation** more intuitive and readable.

```
let a = 5;
let b = 10;
let result = `The sum of ${a} and ${b} is ${a + b}.`;
// result is "The sum of 5 and 10 is 15."
```

Destructuring

Destructuring in JavaScript is a powerful feature that allows you to extract values from arrays or properties from objects and bind them to variables in a concise and expressive way. It provides a convenient syntax for extracting data from arrays and objects.

Array Destructuring:

1. We can destructure values individually
2. We can can skip elements in the array by using commas.
3. If the value at the specified position is undefined, you can assign a default value.
4. We can use rest operator (...) to collect the remaining elements into a new array.
5. We can destructure nested arrays

const numbers = [1, 2, 3, 4, 5];

const [first, second, ...rest] = numbers; // Extracting values from the array into variables

console.log(first); // Output: 1

console.log(second); // Output: 2

console.log(rest); // Output: [3, 4, 5]

Object Destructuring:

1. We can destructure values individually
2. We can rename variables while destructuring by using a colon (:).
3. We can set default values for properties that might be undefined.
4. Destructuring can be used to extract values from nested objects.
5. We can pass remaining values using rest operator

```
const person = { name: 'John', age: 30, city: 'New York' };

const { name, age } = person; // Extracting properties from the object into variables

console.log(name); // Output: 'John'

console.log(age); // Output: 30
```

Nested objects destructuring

```
const person = { name: 'John', age: 30, address: { city: 'New York', country: 'USA' } };

const { name, address: { city, country } } = person; // Nested destructuring

console.log(name); // Output: 'John'

console.log(city); // Output: 'New York'

console.log(country); // Output: 'USA'
```

Spread Syntax:

spread operator (...) in JavaScript is a powerful tool used to expand elements of an iterable (like arrays, strings, or objects) into individual elements. It's often used for array manipulation, function arguments, and object spreading.

Spread operator is used to spread the data

which is present in array and string

Spread operator is used as function argument

Spread operator will create deep copy for array and object

Spread is denoted by ...

for array [...arrRef] , for object {...objectRef}

Expanding an Array:

```
const arr1 = [1, 2, 3];
const arr2 = [...arr, 4, 5, 6];
console.log(arr2); // Output: [1, 2, 3, 4, 5, 6]
```

Passing Arrays as Function Arguments:

```
const numbers = [1, 2, 3, 4, 5];
const sum = (a, b, c, d, e) => a + b + c + d + e;
console.log(sum(...numbers)); // Output: 15
```

Merging Objects:

```
const obj1 = { name: 'john' };
const obj2 = { age: 23 };
const mergedObj = { ...obj1, ...obj2 };
console.log(mergedObj); // Output: { name: 'john', age: 23 }
```

shallow copy with the example

```
const originalArray = [1, 2, 3, 4, 5];

const shallowCopyArray = [...originalArray]; // Shallow copy using the rest operator

shallowCopyArray[0] = 10; // Modify the shallow copy

console.log(originalArray); // Output: [1, 2, 3, 4, 5]

console.log(shallowCopyArray); // Output: [10, 2, 3, 4, 5]
```

Why spread operator introduced

```
var a=[12,2,2,22,]  //create an array

var b=a;  //assigned a array to another variable

b[0]=500; //change the second variable value

 console.log(a);//500,2,2,22
```

-here value of array a is also changed because here value is not assigned . memory location is assigned in order to overcome this spread operator was introduced

Rest Parameters:

Rest operator is introduced in ES6

Rest operator are used in function parameters

Rest operator is denoted by ...

Rest operator is implicitly an array

Rest operator will accepts from 0 to n number of arguments

Rest operator should be the last parameter of function

In Functions parameters rest operators can be only one

Rest Operator can be used in array/ object destructuring

Rest Operator is used for better readibility

Syntax :

...varName (passed as function parameter)

Allows representing an indefinite number of arguments as an array.

```
function sum(a,b,...args) {
   return …args
}
```

sum(1,2,3,4,5,6,7)