**Let vs var**

Var global scope me hota means ap isko kisi bhi function me use kr sakte aik bar declare kara kr or outside of functionbhi use kr sakte.

Let blocked scope hai means ap isko srf ussi function me use kr sakte jis me banaya or outside of function nhi use kr sakte.

Var ko empty rakha sakte example: var a ;

Let ko empty nhi rakh sakte example: let a; //error

**Const:**

Const means jo bhi variable ki value hai use hum dobara likh kr change nhi kr sakte or ye srf readonly ke liye use hota

Or is me object or array bhi use kr sakte

Isse liye ES6 me const use hota zayda tar

Summary

* The const keyword creates a read-only reference to a value. The readonly reference cannot be reassigned but the value can be change.
* The variables declared by the const keyword are blocked-scope and cannot be redeclared.

**Default function parameters**

function say(message='Hi') {

console.log(message);

}

say(); // 'Hi'

say(undefined); // 'Hi'

say('Hello'); // 'Hello'

How it works.

* In the first function call, we didn’t pass any argument into the say() function, therefore message parameter took the default value 'Hi'.
* In the second function call, we passed the undefined into the say() function, hence the message parameter also took the default value 'Hi'.
* In the third function call, we passed the 'Hello' string into the say() function, therefore message parameter took the string 'Hello' as the default value.

[**Rest parameter**](https://www.javascripttutorial.net/es6/javascript-rest-parameters/)

Agr apko pata naik e user kitne argument yak is type ke argument use kreyga to ap rest parameter ka use kr sakte

Example1:

function fn(...args) {

//...

}

fn(1, 2, 3, "A", "B", "C");

sb arguments print hongy NOTE: params me args name use kara means args ab object hai or usme wo sb elements agay.

example2:

function fn(a,b,...args) {

//...

}

fn(1, 2, 3, "A", "B", "C");

means phely two arguments nhi ayengy baqi sb ayngy

example3:

function fn(a,...rest, b) {

*// error*

}

SyntaxError: Rest parameter must be last formal parameter

Means rest parameter initialize krwane ke baad koi or param nhi dey sakte

Example 4:

function sum(...args) {

return args

.filter(function (e) {

return typeof e === 'number';

})

.reduce(function (prev, curr) {

return prev + curr;

});

}

let result = sum(10,'Hi',null,undefined,20);

console.log(result);

agr apko srf aik type ka parameter chaheye e.g number to ye func use kr saqkte

**spread operator:**

The spread operator allows you to spread out elements of an iterable object such as an array, map, or set.

Means array ke elements ko spread kr deyga matlab jo elements array me thy wo ab array me nhi rahy

For example1:

const odd = [1,3,5];

const combined = [2,4,6, ...odd];

console.log(combined);

Code language: JavaScript (javascript)

Output:

[ 2, 4, 6, 1, 3, 5 ]

Here are the main differences:

* The spread operator (...) unpacks the elements of an iterable object.
* The rest parameter (...) packs the elements into an array.

Multiple elements ko array me dalta rest parameter

Or array ko tor kr elements bahir nikal ta spread operator

Example2:

The rest parameters must be the last arguments of a [function](https://www.javascripttutorial.net/javascript-function/). However, the spread operator can be anywhere:

const odd = [1,3,5];

const combined = [...odd, 2,4,6];

console.log(combined);

Code language: JavaScript (javascript)

Output:

[ 1, 3, 5, 2, 4, 6 ]

Example3:

function compare(a, b) {

return a - b;

}

let result = compare(...[1, 2]);

console.log(result); // -1

Consider the following example:

let chars = ['A', ...'BC', 'D'];

console.log(chars); *// ["A", "B", "C", "D"]*

Code language: JavaScript (javascript)

In this example, we constructed the chars array from individual strings. When we applied the spread operator to the 'BC'string, it spreads out each individual character of the string 'BC' into individual characters.

Summary

* The spread operator is denoted by three dots (…).
* The spread operator unpacks elements of iterable objects such as arrays, sets, and maps into a list.
* The rest paramter is also denoted by three dots (…). However, it packs the remaining arguments of a function into an array.
* The spread operator can be used to clone an iterable object or merge iterable objects into one.

# **Object Literal Syntax Extensions in ES6**

Agr object ke andar kesi key ka name wohi ho jo value ka name hai to ap object literal use kr sakte

Example:

function createMachine(name, status) {

return {

name: name,

status: status

};

}

Ab is me do do bar name or status araha to confuse ho rhy

function createMachine(name, status) {

return {

name,

status

};

}

is liye object literal use krte tky agr key value ka name same hai to srf aik bar likho

Prior to ES6, you could use the square brackets( [])  to enable the **computed property names** for the properties on objects.

The square brackets allow you to use the string literals and variables as the property names.

See the following example:

example3:

let name = 'machine name';

let machine = {

[name]: 'server',

'machine hours': 10000

};

console.log(machine[name]); // server

console.log(machine['machine hours']); // 10000

example4:

let server = {

name: "Server",

restart: function () {

console.log("The" + this.name + " is restarting...");

}

};

“this” keyword use for ke iss object me jo key hai uski value ley ao

# **JavaScript for…of Loop**

Jese for loop hota wesa hi hai lekin limit nhi ke kahan tk chalna

let scores = [80, 90, 70];

for (let score of scores) {

score = score + 5;

console.log(score);

}

Array ke elements ko new variable me store krta

Agr ap score ki value change nhi krna chahte to ap const use krlo magr +5 hata deyna warna error ayga

Example2:

let colors = ['Red', 'Green', 'Blue'];

for (const [index, color] of colors.entries()) {

console.log(`${color} is at index ${index}`);

}

output

Red is at index 0

Green is at index 1

Blue is at index 2

for...of vs. for...in

The for...in iterates over all [enumerable properties](https://www.javascripttutorial.net/javascript-enumerable-properties/) of an object. It doesn’t iterate over a collection such as Array, Map or Set.

Unlike the for...in loop, the for...of iterates a collection, rather than an object. In fact, the for...of iterates over elements of any collection that has the [[Symbol.iterator]](https://www.javascripttutorial.net/es6/symbol/#iterator) property.

Means for in loop ke andar ko variable banaya for index return krta or for of loop values return krta example:

let scores = [10,20,30];

scores.message = 'Hi';

console.log("for...in:");

for (let score in scores) {

console.log(score);

}

console.log('for...of:');

for (let score of scores) {

console.log(score);

}

Output:

for...in:

0

1

2

message

for...of:

10

20

30

In this example, the for…in statement iterates over the properties of the scores array:

while the for…of iterates over the element of an array:

**array destructuring**

simple def: array ke elements ko hum varaiable me store krwana chahte usko array destrucring khete.

Yahan destructuring index ke hissab se hoti

Example: fruits = [ ‘aple’, ‘banana’, ‘mango’]

Now its destructuring

Let [ fruitx, fruity, fruit Z ] = fruits

Console.log(fruitx)// apple

Ab agr mujhe fruits ki 2 value nhi chahye to me aese likhonga

Let [ fruitx, , fruit Z ] = fruits

Or agr koi fourth value bhi nikalne ki kohish kre to ap default value deysakte fruitP=”default”

Or agr apko nhi pata ke array me kitne elemnts hai to ap rest operator use kro

Let [ …fruitx] = fruits

## object destructuring

simple define: object me jo key li values hoti unko aik variable me store krnwane ko object dest

yahan object ke andar jo key hai uspr destructuring krte

Example: fruits = { name:”hammad”, lastname:”ayub”,email:”hammad@gml”}

Now its destructuring

Let {email}= fruits

Console.log(email)// hammad@gml

Baqi same wohi working jo upr kri

Default values bhi deysakte

Or isme rest parameter use kr sakte

fruits = { name:”hammad”, lastname:”ayub”,email:”hammad@gml”}

Let {…email}= fruits

Sb key values dey deyga

let { firstName, lastName } = getPerson() || {};

Now, no error will occur. And the firstName and lastName will be undefined.

Export and import module

Wese to export import wesi krte jese kr rhy baqi kuch important dekhlo

### **Import an entire module as an object**

To import everything from a module as a single object, you use the asterisk (\*) pattern as follows:

import \* as cal from './cal.js';

cal alias name hai ab iska use krke dusri file me jo variables banay usko use kr sakte

cal.sum etc

## Limitation of import and export statements

Note that you must use the import or export statement outside other statements and functions. The following example causes a SyntaxError:

if( requiredSum ) {

export sum;

}

## Re-exporting a binding

It’s possible to export bindings that you have imported. This is called re-exporting. For example:

import { sum } from './math.js';

export { sum };

**CLASS:**

PORANA METHOD:

jese python oop me hota tha aik class banai usme aik constructor banaya or bhot sary functions bana diye yahan js me class nhi banti baqi nechy example me jo Person hai wo as a constructor hai or jo protype banai wo functions hai

ab us constructor ka object banaya call krke or phr uska instance bana diya

Prior to ES6, JavaScript had no concepts of classes. To mimic a class, you often use the [constructor/prototype pattern](https://www.javascripttutorial.net/javascript-constructor-prototype/) as shown in the following example:

function Person(name) {

this.name = name;

}

Person.prototype.getName = function () {

return this.name;

};

var john = new Person("John Doe");

console.log(john.getName());

Output:

John Doe

ES6 method

## ES6 class declaration

ES6 introduced a new syntax for declaring a class as shown in this example:

class Person {

constructor(name) {

this.name = name;

}

getName() {

return this.name;

}

}

var john = new Person("John Doe");

console.log(john.getName());

**symbol**