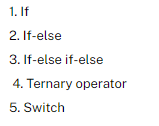
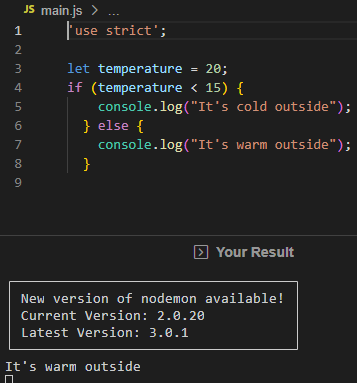
**MODULE – 4 CONDITIONAL EXECUTIONS**

**The below all can be used to write conditional statements:**

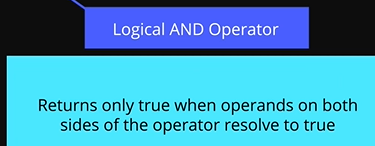


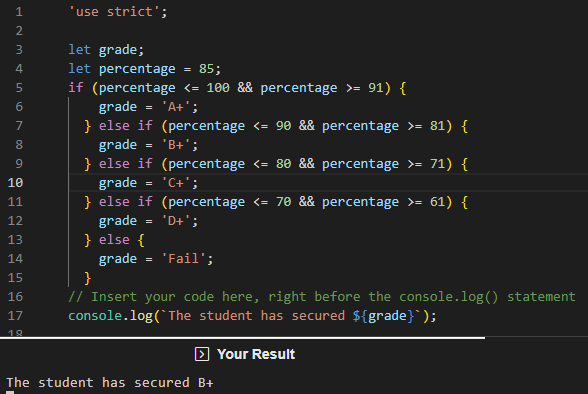
**if-else:**



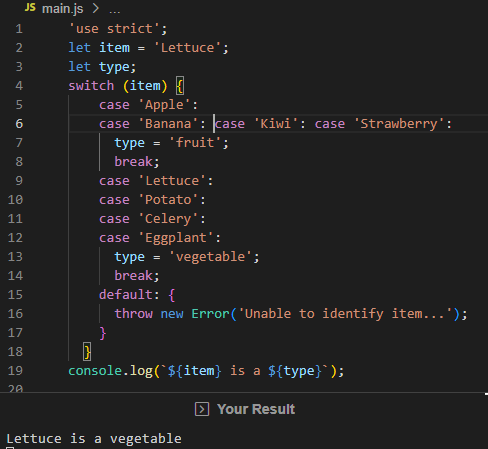
**if else-if else:**

**&&**

****



**Switch:**



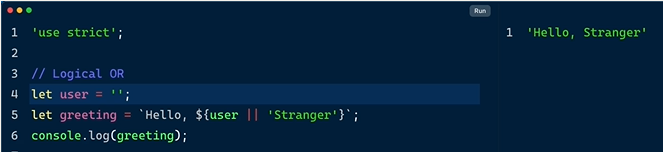
**Ternary Operator:**



**Binary Logic Operators:**

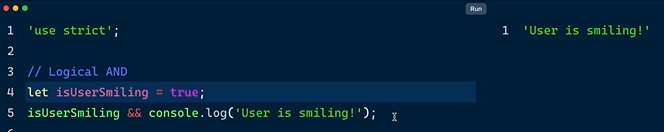
**Logical OR operator:**

* User is falsie
* || pipe character is used for or operator
* User name pedithe becomes truthy and prints greeting with user name
* User name Em pettaka pothe apudu or operator work ayyi stranger ani print avthundi

****

**Logical AND operator:**

* Variable false ayithe output kuda false ey
* Short syntax edi

****

**Logical AND assignment operator:**

* &&=
* Line 7 syntax
* Data downloaded and strict operator 100 ayithe data downloaded now complete ani vasthundi

****

**Logical AND and OR assignment operator:**

* ||
* Pipe character line 8
* Datadownloaded 100 kakapothe
* Downloading ani print avthundi

****

**SUMMARY:**

* Logical and operator
* Logical or operator
* Logical and assignment
* Logical or assignment
* Nullish coalescing operator
* Optional chaining operator
* Binary logic operator
* Provides shortcut syntax to express complex conditional statements

****

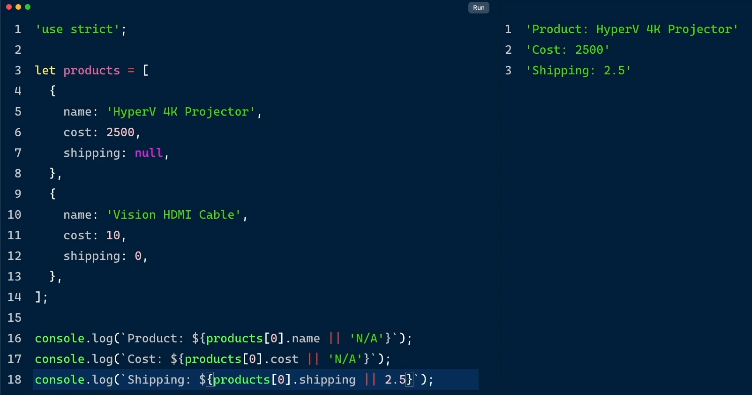
* And operator &&
* Or operator ||
* Logic and &&=
* Logic or ||=



* ??
* Similar to or operator
* Replace || with ??

**Array featuring 2 objects:**

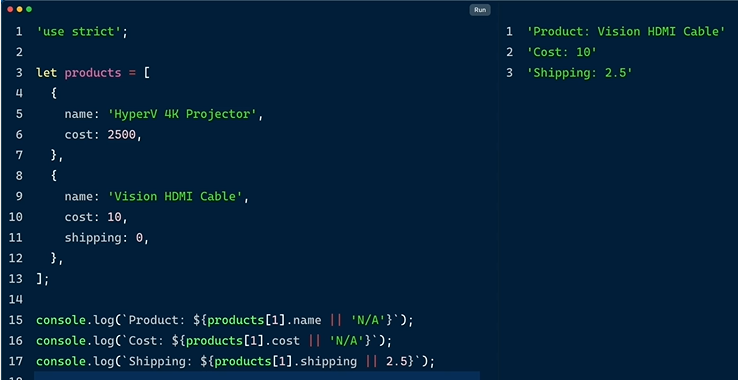
* Displaying name in object products zero index
* Displaying cost in object products zero index
* Displaying shipping in object products zero index which is null or operator undhi so 2.5 print ayindi
* Or operator absence of data
* Vendor nunchi shipping cost null undi manam 2.5 ani update chesam or operator tho
* Object lo asalu shipping property lekapoyina e syntax work avthundi

****

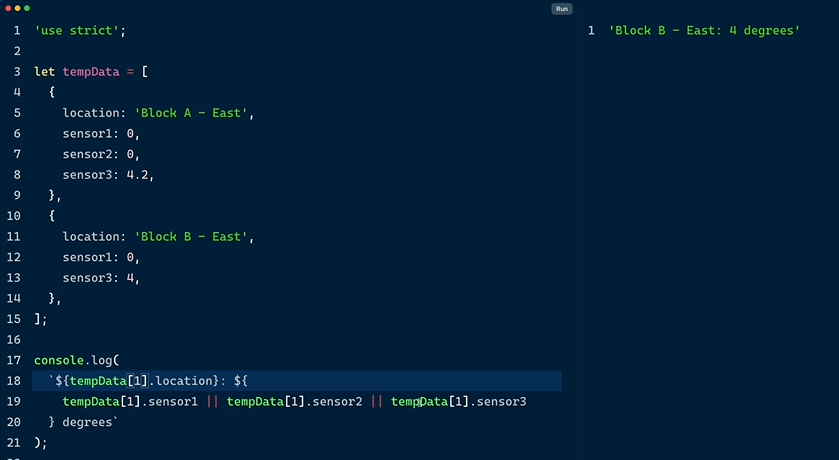
**Now access index one:**

* Index 1 lo shipping 0 ani undhi kani or operator valla 2.5 ani update avthundi
* E problem solve avvali ante Nullish coalescing operator use cheyali ?? instead of ||

** **

****

**Or operator || property 0 unte read cheyadu ?? Nullish coalescing operator read chesthundi**

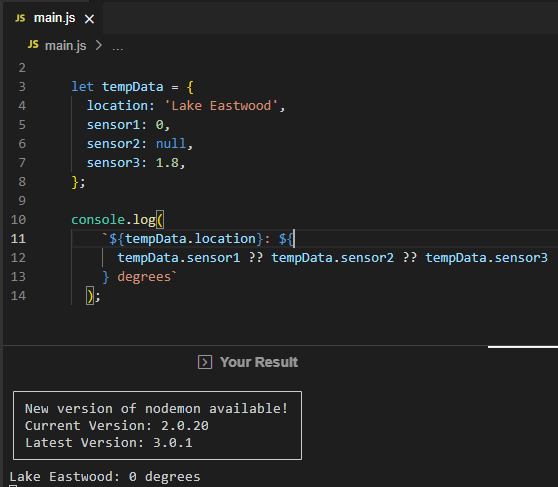
****

* || and operator 0 ni read cheyatledu
* Index 1 lo 0 read cheyakunda escape ayyi 4 ni read chesindi

****

* ?? 0 ni read chesthundi

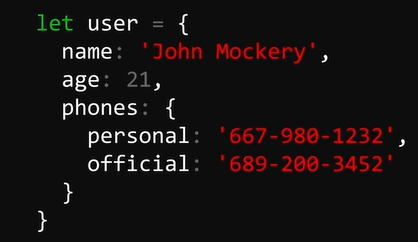
**EXERCISE:**

****

****

**Nested Object:**

* User ane object lopala
* Phones ane object create chesam
* E properties ni access cheyali ante 2 ways untayi
* Bracket syntax
* Dot syntax

****

Dot syntax use chesi user obj lo phone ob nunchi personal number ni access chesam

****

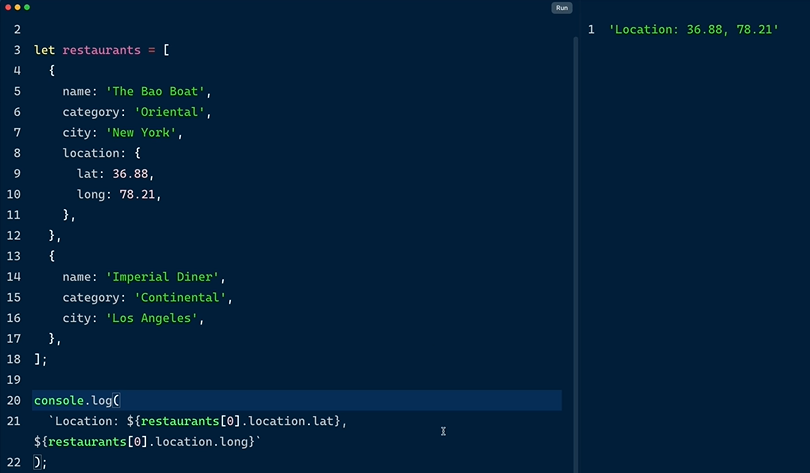
E dot syntax tho problem enti ante address city 2 properties obj lo levu

Non existing properties ni access cheyalemu

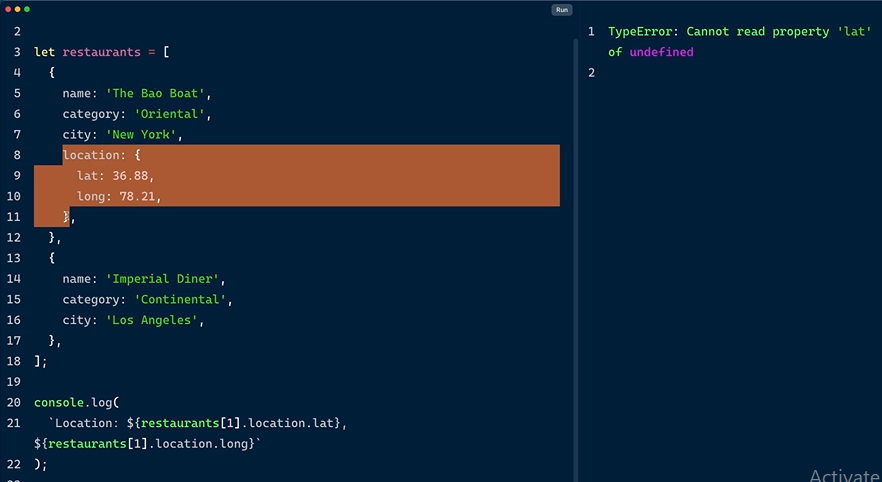
****

? tho replace cheyali instead of dot error rakunda undefined ani vasthundi

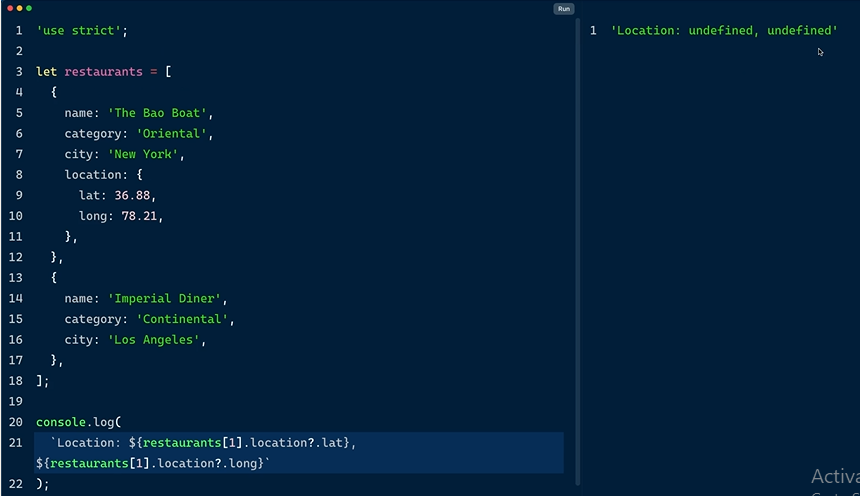
****



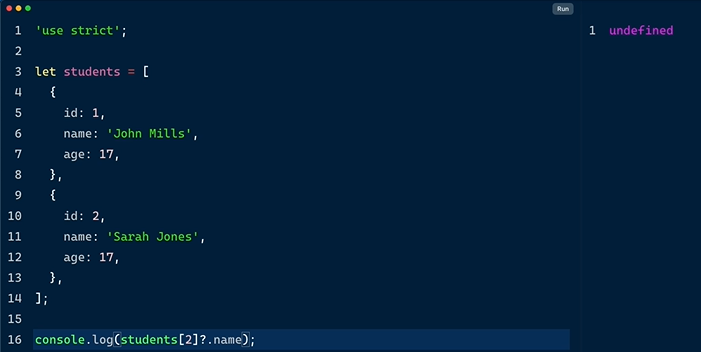
* Dot tho property ni access from index 0 chesam works perfectly



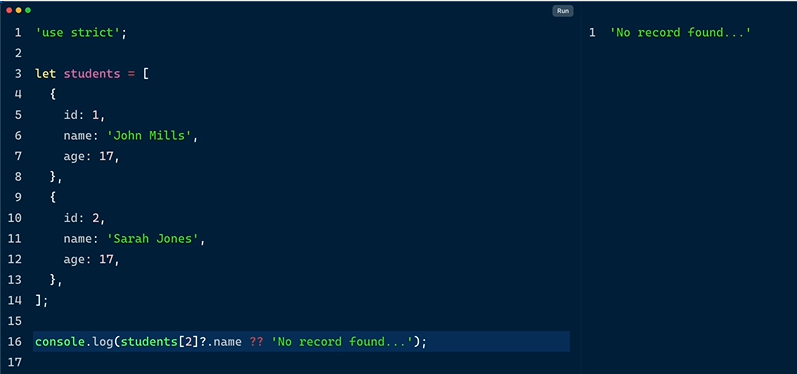
* Index 1 lo location attribute ey ledu
* So type error ani vachindhi deni overcome cheyali
* Error kakunda undefined ani chupinchali



* Dot mundu ? pettali
* Location unda ani read chesi lekapothe undefined ani print cheyi



* Index 2 lo name property unda read cheyi
* Lekapothe ? optional chnaing operator undefined ani print cheyi



* Student obj lo 2nd index lo name property unda read cheyi
* Lekapothe Nullish coalescing operator ?? use chesi no record found ani print cheyi

**EXERCISE:**

'use strict';

const users = [

  {

    name: 'Joe',

    address: {

      home: '212, River Drive',

      city: 'New York',

      country: 'US',

    },

  },

  {

    name: 'Jane',

  },

];

try {

  console.log(`${users[0].name} lives in ${users[0].address.city}`);

  console.log(`${users[1].name} lives in ${users[1].address.city}`);

} catch (error) {

    console.log(

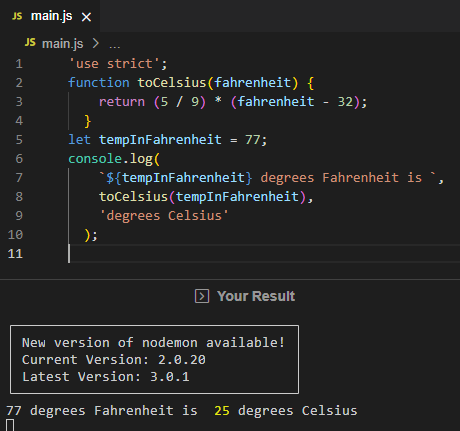
        `${users[1].name} lives in ${users[1].address?.city ?? 'a big city'}`

      );

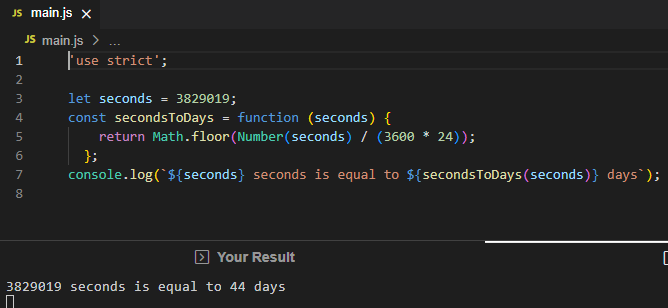
}

**MODULE – 5 FUNCTIONS ESSENTIALS & MORE**

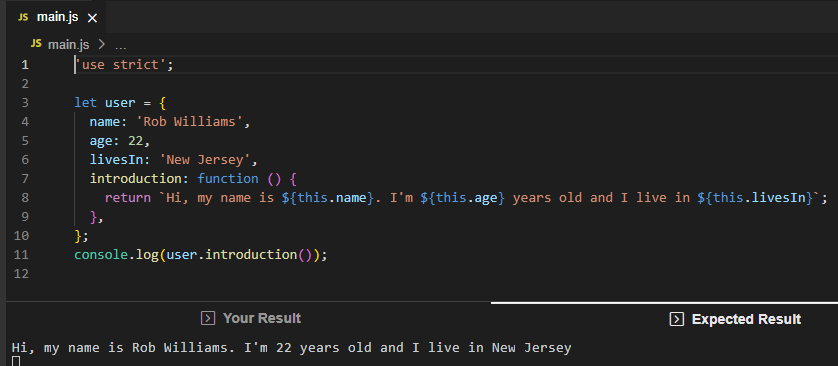
**** Return keyword in functions

****

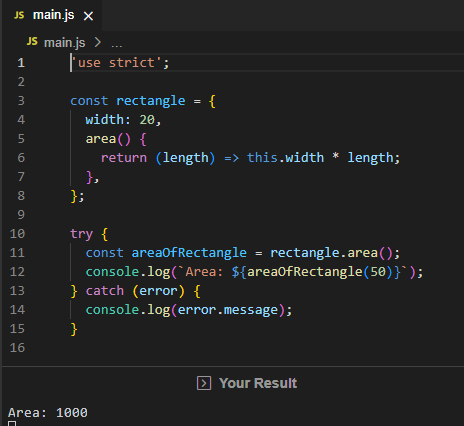
Function expression



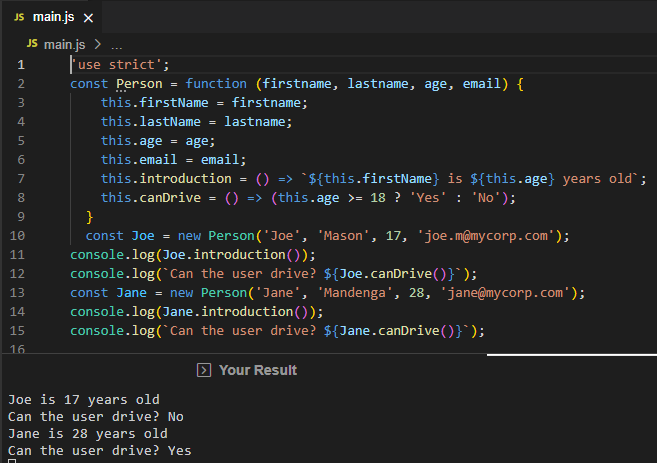
This keyword & function execution context



Arrow functions



Constructor functions



Default parameters

'use strict';

const computeReturns = function (principal, rateOfInterest = 8, tenure = 10) {

  return principal + (principal \* rateOfInterest \* tenure) / 100;

};

const investments = {

  joe: {

    p: 20000,

    t: 30,

  },

  jane: {

    p: 45000,

    r: 10.2,

    t: 20,

  },

  mike: {

    p: 54000,

  },

};

console.log(

  `Joe invested ${investments.joe.p} and got back: `,

  computeReturns(investments.joe.p, investments.joe.r, investments.joe.t)

);

console.log(

  `Jane invested ${investments.jane.p} and got back: `,

  computeReturns(investments.jane.p, investments.jane.r, investments.jane.t)

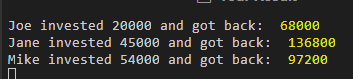
);

console.log(

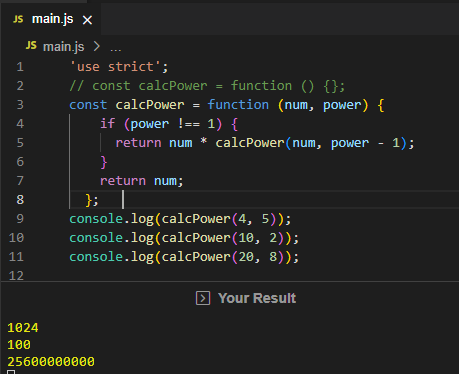
  `Mike invested ${investments.mike.p} and got back: `,

  computeReturns(investments.mike.p, investments.mike.r, investments.mike.t)

);



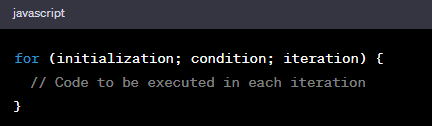
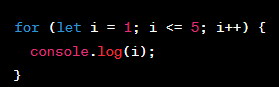
RECURSIVE FUNCTIONS

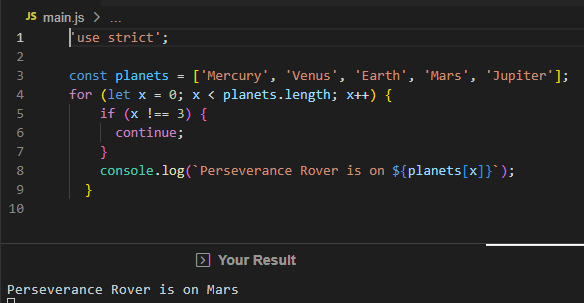


**MODULE – 6 LOOPS**

**FOR LOOP**

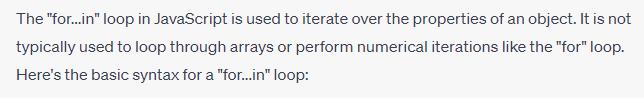
**Syntax:**

** **

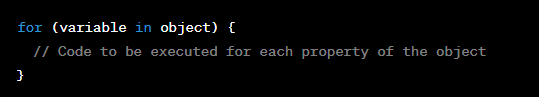
****

**FOR IN FOR OF**

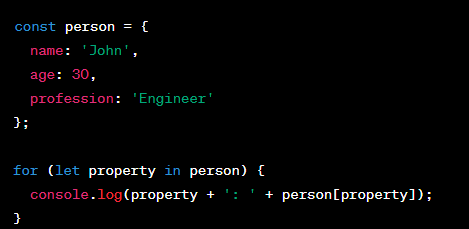
**For in**

****

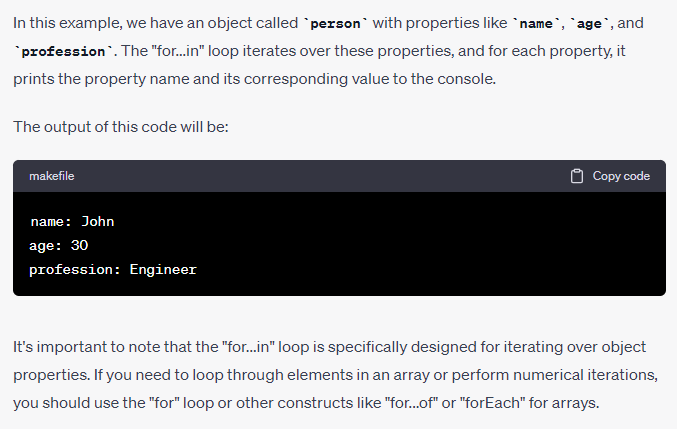
**Syntax:**

****

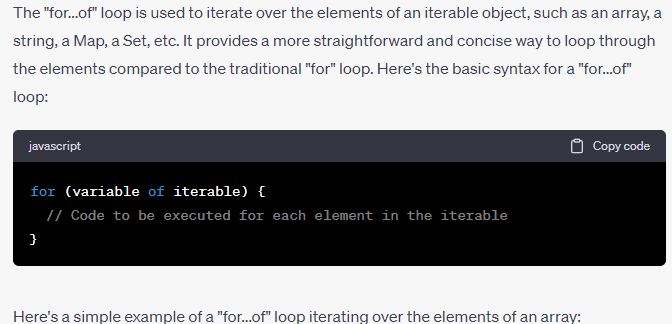
**Example:**

****

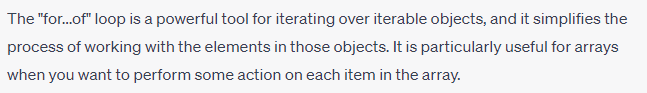
**EXPLANATION:**

****

**FOR OF LOOP:**

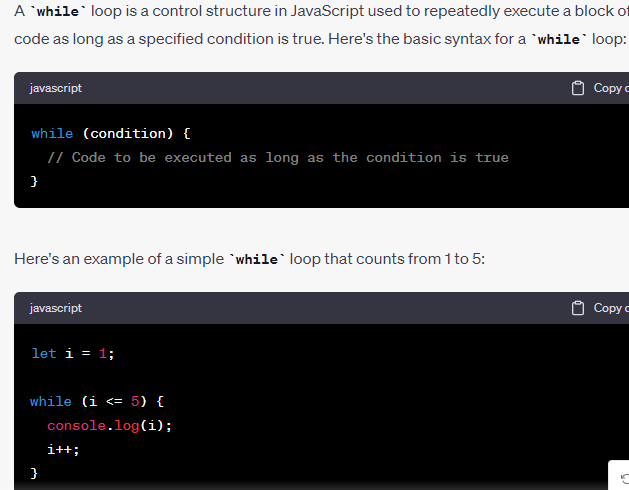
****

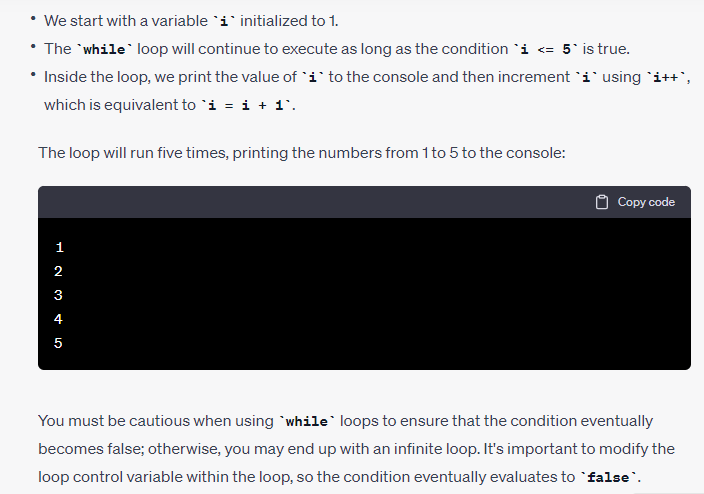
****

****

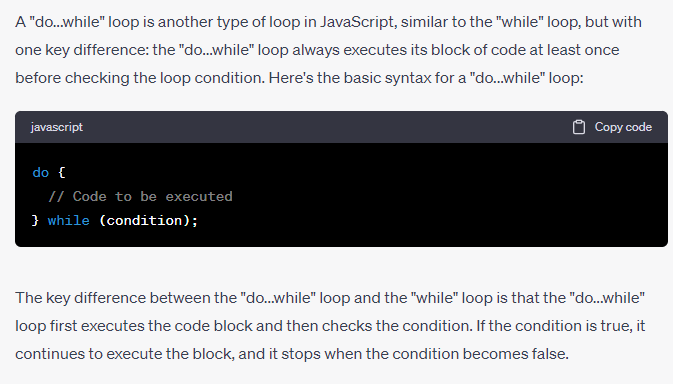
**While – do while loop**

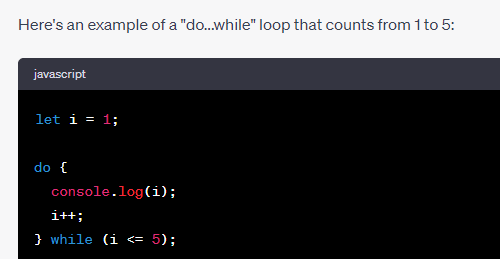
**While loop:**

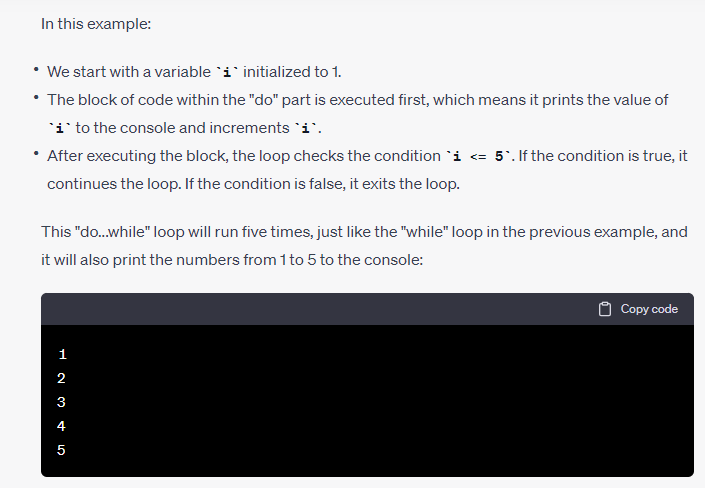
****

****

**DO-WHILE-LOOP:**

****

****

****

The "do...while" loop is useful when you want to ensure that a block of code runs at least once, regardless of the initial condition.

'use strict';

let m = 0,

  n = 1,

  evenNumbers = [],

  oddNumbers = [];

// Use the while loop here

while (m <= 100) {

    evenNumbers.push(m);

    m += 2;

  }

// Use the do-while loop here

do {

    oddNumbers.push(n);

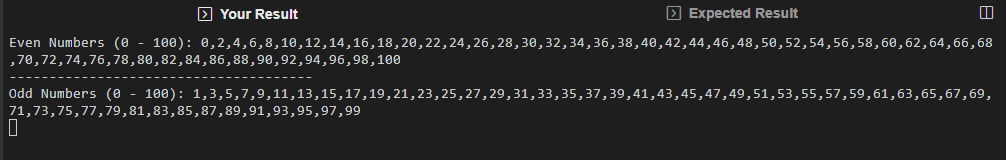
    n += 2;

  } while (n <= 100);

console.log(`Even Numbers (0 - 100): ${evenNumbers.toString()}`);

console.log('--------------------------------------');

console.log(`Odd Numbers (0 - 100): ${oddNumbers.toString()}`);

****