Ex:- // module.js

```
export const add = (a, b) => a+b;
```

// main.js

```
import {add} from './module.js';
console.log(add(2,3));
```

5 marks:-

1) differentiate types of variable in js with a truth table.

Ans

1) var :- var is function-scoped. It means that variables declared with 'var' are accessible within the function they are declared in, regardless of block scope.

- variables declared with 'var' can be redeclared and reassigned within the same scope.

Syntax:- var variableName = value;

Ex:-

```
var x = 10;
function example() {
    var x = 20;
    console.log(x); // output 20
}
example();
console.log(x); // output : 10
```

Page:
Date: / /

(2) let :- 'let' is block-scoped. It means that variables declared with 'let' are accessible only within the block they are declared in (e.g., within curly braces '{}').

- variables declared with 'let' can be reassigned but cannot be redeclared within the same scope.

syntax :- let variableName = value;

Ex :-
```
let x = 10;
if (true) {
    let x = 20;
    console.log(x); // output 20
}
console.log(x); // output : 10
```

(3) const :- const is also block-scoped like 'let'.

- variables declared with 'const' cannot be reassigned once they are initialized. However, for objects and arrays, the properties or elements of the object or array can be mutated.

- variables declared with 'const' cannot be redeclared within the same scope.

syntax :- const variableName = value;

Ex :-
```
const PI = 3.14;
PI = 3.14159 // cannot be reassigned
const person = { name: 'Rahul' };
person.age = 20; // properties of object can be modified is valid
```

Page:
Date: / /

Truth Table :-

Behavior	Var	let	const
Hoisting	Hoisted	Not Hoisted	Not Hoisted
Block scope	No	yes	yes
can be reassigned	yes	yes	no
can be re-declared	yes	no	no
function scope	yes	yes	yes

2) Explain template literal with ternary operator. give an example.

Ans Template literals are a feature in Javascript that allow for easier string interpolation and multiline strings by using backticks (`) instead of single or double quotes.

- They also support expression interpolation, including the use of ternary operators.
- The ternary operator (`? :`) is a concise way to write conditional statements in Javascript
- it evaluates a condition and returns one value if the condition is true, and another value if the condition is false.
- Combining template literals with the ternary operator allows ye to dynamically construct strings based on conditions.

Page:
Date: / /

- A ternary operator is a concise way to write conditional statements. It has the following syntax:-

'condition ? expressionIfTrue : expressionIfFalse'.

Ex:-

```
const temperature = 25;
const weather = temperature > 20 ?
                'warm' : 'cold';

const message = 'The weather today is
${weather} with a temperature of ${temperature}
degrees celsius.';

console.log(message)
```

Page:
Date: / /

3) Explain async await as asynchronous javascript with example.

Ans Async/await is modern way of handling asynchronous in Javascript. It provides a more readable and structured approach to writing asynchronous code compared to using callbacks or promises.

Async function:
Async simply allows us to write promises based code as if it was synchronous and it checks that we are not breacking the execution thread.
- Async functions will always return a value.

Await function:- Await function is used to wait for the promise. It could be used within the async block only.
- It makes the code wait until the promise returns a result.

- The async keyword transforms a regular javascript function into an asynchronous function.

- The await keyword is used inside an async function to pause its execution and wait for a promise to resolve before continuing.

Page:
Date: / /

Ex:-
```
function fetchData () {
    return new Promise ((resolve, reject) => {
        setTimeout (() => {
            resolve ('Data fetched successfully!');
        }, 2000);
    });
}

async function ~~fetch~~ DataAsync () {
    try {
        console.log ('fetching data...');
        const data = await fetchData ();
        console.log ('Data received:', data);
    } catch (error) {
        console.log ('Error', error);
    }
}
DataAsync ();
console.log ('Async operation started...');
```

Page:
Date: / /

4] give an example of real time entity problem using oop is class and object.

Ans example of a Bank account using oop.

Class :- BankAccount

- attributes :-
  - accountNumber : ac_Number
  - accountHolder : ac_Holder
  - balance.

- method :-
  - deposit()
  - withdraw()
  - checkBalance()

Example :-

```
Class BankAccount {
    constructor (ac_no, ac_Holder, balance=0){
        this.ac_no = ac_no;
        this.ac_Holder = ac_Holder;
        this.balance = balance;
    }

    deposit (amount) {
        this.balance += amount;
        console.log(`Deposit of ${amount}
        successful. current balance : ${this.balance}`
        );
    }
```

```
withdraw(amount) {
  if(amount > this.balance){
    console.log("Insufficient funds in account");
  }else
  {
    this.balance -= amount;
    console.log(`withdrawal of ${amount}
    successful. current balance : ${this.balance}`
    );
  }
}
checkBalance(){
    console.log(`current balance : ${this.
        balance}`);
}
}
const account1 = new BankAccount(12345, 'Rahul');
account1.deposit(1000000);
account1.withdraw(500000);
account1.getBalance();
```

Page:
Date: / /

4 mark module:-1

3) give appropriate example of map, filter, for of and for in with concept.

Ans 1) map() :- The 'map()' function in JavaScript is used to apply a function to each element of an array and creates a new array with the results of calling that function on each element.

- it doesn't mutate the original array.

Syntax:-

```
map((element, index, array) => { /*...*/ })
```

Ex:- const number = [1,2,3,4,5];

```
const squrno = number.map(num => num*num)
console.log(squrno);
```

2) filter :- The 'filter()' function in JavaScript is used to filter out elements from an array based on a specified condition.

- It creates a new array containing only the elements that pass the condition provided by a callback function.

Syntax:-

```
array.filter(callback(element, index, arr), this value)
```

Page:
Date: / /

Ex:- const ages = [32, 30, 16, 40];

function check(age) {
return age >= 18;
}
const result = ages.filter(check);
console.log(result);

3) for... of :- the for... of loop in JavaScript is used to iterate over iterable objects like arrays, strings, maps, sets, etc.
It provides a concise and readable way to loop through the elements of an iterable without explicitly tracking the index.

Syntax:- for (variable of iterable) {
// code
}

Ex:- const fruits = ['apple', 'banana', 'kiwi'];

for (const fruit of fruits) {
console.log(fruit);
}

4) for... in :- for... in loop in JavaScript is used to iterate over the properties of an object.

Page:
Date: / /

- It iterates over all enumerable properties of an object, including inherited enumerable properties from its prototype chain.

Syntax:

```
for (variable in object){
    // code
}
```

Ex:-

```
const Person = {
    name : 'Rahul',
    age : '21',
    city : 'Jamnagar'
};
for (let key in Person)
{
  console.log(`${key} : ${Person[key]}`);
}
```