1.(a). Create a set of JavaScript code examples that demonstrate the use of arithmetic, assignment, comparison operators and provide scenarios.

Arithmetic operations:

let sum = 5 + 3;

let difference = 10 - 4;

let product = 2 * 6;

let quotient = 20 / 5;

let remainder = 15 % 4;

- Arithmetic Operators:
- Assignment Operators:
- Comparison Operators:

Assignment Operators: Commonly used in loops, where you might want to increment a counter, or in event handling to update variables.

let x = 10;

let y = 5;

x += 3;

y -= 2;

x *= 2;

y /= 2;

x %= 5;

Comparison Operators: Useful in conditions like if statements, loops, and ternary operators to make decisions based on the relationship between variables. For example, checking if a user input is within a certain range, or if one value is greater than another in a sorting algorithm.
let a = 10;

let b = 5;

console.log(a == b);

console.log(a===b);//Its for datatypetype checking

console.log(a != b);

console.log(a > b);

console.log(a < b);

console.log(a >= b);

console.log(a <= b);

Logical Operators:

let x = true;

let y = false;

console.log(x && y);

```
console.log(x || y);

console.log(!x);
```

Scenarios:

- Consider a website where users need to be logged in to access certain features. You can use logical AND (&&) to check if both the user is logged in and has the required role to access the feature.
- In a form validation script, you might use logical OR (||) to ensure that at least one of the required fields is filled out.
- You can use logical NOT (!) to toggle a boolean value, for example, switching a light on/off in a smart home system.

Bitwise Operators:

```
let a = 5;

let b = 3;


console.log(a & b);

console.log(a | b);

console.log(a ^ b);

console.log(~a);

console.log(a << 1);

console.log(a >> 1);
```

Scenarios:

- In cryptography, bitwise operators are widely used for encoding and decoding data.
- Bitwise operators are also used in low-level programming, such as device drivers, where you may need to manipulate individual bits of data.
- In graphics programming, bitwise operators can be used for pixel manipulation and image processing.

Ternary Operator (Conditional Operator):

```
let age = 20;

let can = age >= 18 ? "Yes" : "No";

console.log(can);
```

- In a user interface, you might use the ternary operator to conditionally render elements based on user permissions or settings.
- When processing user input, you can use the ternary operator to validate and handle different cases efficiently.
- In a game, you might use the ternary operator to determine whether a player has achieved a certain score and display a corresponding message.

1. (b) Implement a "To-Do List Manager" with JavaScript Objects where each task in the to-do list should be represented as an object with properties like task description, due date, and completion status.

Source code :

```
let todoList = [];

function addTask(description, dueDate) {
   let newTask = {
      description: description,
      dueDate: dueDate,
      completed: false
   };

   todoList.push(newTask);
}

function completeTask(index) {
   if (index >= 0 && index < todoList.length) {
      todoList[index].completed = true;
   } else {
      console.log('Invalid task index.');
   }
}

function displayTodoList() {
   console.log('To-Do List:');
   todoList.forEach((task, index) => {
      console.log(`${index + 1}. [${task.completed ? 'X' : ' '}] ${task.description} - Due: ${task.dueDate}`);
   });
}

addTask('Complete project proposal', '2024-04-01');
addTask('Buy groceries', '2024-03-28');
addTask('Call mom', '2024-03-30');

completeTask(1);

displayTodoList();
```

2. (a). Design a JavaScript object to represent a library catalog where each book in the catalog should be represented as a separate object with properties like title, author, publication-Date, publisher, availability ISBN etc.

Source code :

```
let libraryCatalog = {
   books: [],
   addBook: function(title, author, publicationDate, publisher, availability, ISBN) {
      this.books.push({
         title: title,
         author: author,
         publicationDate: publicationDate,
         publisher: publisher,
         availability: availability,
         ISBN: ISBN
      });
```

```
    },

    removeBook: function(ISBN) {
        this.books = this.books.filter(book => book.ISBN !== ISBN);
    },

    findBookByISBN: function(ISBN) {
        return this.books.find(book => book.ISBN === ISBN);
    },

    displayCatalog: function() {
        console.log('Library Catalog:');
        this.books.forEach(book => {
            console.log(`Anime Title: ${book.title}, Author: ${book.author}, ISBN: ${book.ISBN}`);
        });
    }
};
libraryCatalog.addBook('Naruto', 'Masashi Kishimoto', '1999', 'Shueisha', true, '9784088737345');
libraryCatalog.addBook('Attack on Titan', 'Hajime Isayama', '2009', 'Kodansha', true, '9784063409465');
libraryCatalog.addBook('One Piece', 'Eiichiro Oda', '1997', 'Shueisha', false, '9784088725090');

libraryCatalog.displayCatalog();

// Removing a book
libraryCatalog.removeBook('9784088725090');
console.log('After removing a book:');
libraryCatalog.displayCatalog();
```

2.b .Demonstrate Built-in objects in JavaScript

Object:
The Object object is a core JavaScript data type. It represents key-value pairs, similar to dictionaries in other languages.
It is used to create, manipulate, and interact with JavaScript objects.

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

Array:
The Array object is used to store multiple values in a single variable. It is a special type of object with array-like features.
It provides methods for manipulating arrays, such as push, pop, splice, slice, etc.

```
let fruits = ['Apple', 'Banana', 'Orange'];
```

String:
The String object represents a sequence of characters. Strings are immutable in JavaScript.
It provides methods for manipulating strings, such as charAt, concat, indexOf, slice, toUpperCase, toLowerCase, etc.

```
let message = 'Hello, world!';
```

Number:
The Number object represents numerical data. It is used for mathematical operations.
It provides methods and properties for working with numbers, such as toFixed, toString, isNaN, parseInt, parseFloat, etc.

```
let num = 42;
```

Boolean:
The Boolean object represents a boolean value: true or false.
It provides methods and properties for working with boolean values, although they are rarely used.

```
let isTrue = true;
```

Function:
The Function object is used to define functions.
Functions are first-class citizens in JavaScript, meaning they can be assigned to variables, passed as arguments, and returned from other functions.

```
function greet(name) {
  return 'Hello, ' + name + '!';
}
```

Date:
The Date object represents a specific moment in time.
It provides methods for working with dates and times, such as getDate, getMonth, getFullYear, getHours, getMinutes, etc.

```
let now = new Date();
```

RegExp:
The RegExp object represents a regular expression pattern.
Regular expressions are used for pattern matching within strings.

```
let pattern = /[a-z]+/g;
```

3(a). Designing a Currency Converter Using JavaScript. The application should allow users to input an amount in one currency and convert it to another currency based on the current exchange rate.

```
const exchangeRates = {
  USD: { INR: 74.39 },
  INR: { USD: 0.013 }
};

function convertCurrency(amount, fromCurrency, toCurrency) {
  if (exchangeRates[fromCurrency] && exchangeRates[fromCurrency][toCurrency]) {
    return (amount * exchangeRates[fromCurrency][toCurrency]);
  } else {
    return "Exchange rate not available for the selected currencies.";
  }
}

function currencyConverter() {
  const amount = parseFloat(prompt("Enter amount in USD:"));
  const result = convertCurrency(amount, 'USD', 'INR');
  console.log(`Amount in INR: ${result}`);
}
currencyConverter();
```

3. (b). Develop a JavaScript program that performs client-side validation to validate like email addresses, phone numbers, and passwords.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

4. Develop a basic calculator application using JavaScript which supports arithmetic ooerations like addition, subtraction, multiplication, division etc.

```
function calculate(num1, num2, operator) {

  switch (operator) {

    case '+':

      return num1 + num2;

    case '-':

      return num1 - num2;

    case '*':

      return num1 * num2;

    case '/':

      return num1 / num2;

    default:

      return 'Invalid operator';

  }

}

const num1 = parseInt(prompt("Enter 1st value"));

const num2 = parseInt(prompt("enter 2nd value"));

const operator = prompt("Enter operator in the form(+) :");

const result = calculate(num1, num2, operator);

console.log(result);
```