Assignment-1 JavaScript for JFSD

Objective:

These assignments cover a wide array of JavaScript concepts necessary for Java Full Stack development, encouraging students to apply their understanding of variables, functions, DOM manipulation, asynchronous operations, and object-oriented programming in practical scenarios.

Time:45 mins

Instructions:

1. **Solve the given questions.**
2. **Take screenshot of the code and the executed output**
3. **Upload the above in your respective folder**

JavaScript practice assignment questions that cover essential concepts relevant to Java Full Stack development:

Question 1: Create a JavaScript function that accepts two parameters, adds them together, and returns the result. Test the function by passing different data types (numbers, strings) as arguments.

Coding:

```
// Function to add two values
function addValues(value1, value2)
{
   // Using the "+" operator to add values
   var result = value1 + value2;
   return result;
}

// Test the function with different data types
var sum1 = addValues(5, 10); // Numbers
var sum2 = addValues("Hello", " World"); // Strings
var sum3 = addValues(3.14, 2.5); // Floating-point numbers
```

```
// Display the results
console.log("Sum 1:", sum1);
console.log("Sum 2:", sum2);
console.log("Sum 3:", sum3);
```

Output:

Sum 1: 15

Sum 2: Hello World

Sum 3: 5.640000000000001

Question 2: Develop a program that declares a variable and demonstrates the difference between var, let, and const in terms of scope and assignability.

Coding:

```
// Using var
function varExample()
{
    if (true)
    {
        var x = 10;
        console.log("Inside varExample, x:", x);
    }
    console.log("Outside varExample, x:", x);
}

// Using let
function letExample() {
    if (true) {
        let y = 20;
        console.log("Inside letExample, y:", y);
    }
    // Uncommenting the line below will result in an error because y is not defined here
```

```
    // console.log("Outside letExample, y:", y);
}

// Using const
function constExample()
{
    const z = 30;
    console.log("Inside constExample, z:", z);

    // Uncommenting the line below will result in an error because you cannot reassign a const variable
    }

// Call the examples
varExample();
letExample();
constExample();
```

Output:

```
Inside varExample, x: 10
Outside varExample, x: 10
Inside letExample, y: 20
Inside constExample, z: 30
```

Question 3: Write a JavaScript function that takes an array of numbers and returns the sum of all positive numbers within the array.

Coding:

```
function sumOfPositiveNumbers(numbers)
{
    // Filter out positive numbers
    var positiveNumbers = numbers.filter(function (num)
```

```
    {
        return num > 0;
    });

    // Calculate the sum of positive numbers
    var sum = positiveNumbers.reduce(function (acc, num)
    {
        return acc + num;
    }, 0);

    return sum;
}

// Example usage
var numbersArray = [3, -1, 7, -2, 5, -8, 4];
var result = sumOfPositiveNumbers(numbersArray);

console.log("Sum of positive numbers:", result);
```

Output:

Sum of positive numbers: 19

Question 4: Create a function that generates a random password of a specified length using a combination of uppercase letters, lowercase letters, and numbers.

Coding:

```
function generateRandomPassword(length)
{
    const charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    let password = "";
```

```
    for (let i = 0; i < length; i++)
    {
        const randomIndex = Math.floor(Math.random() * charset.length);
        password += charset[randomIndex];
    }

    return password;
}

// Example usage: Generate a random password of length 12
const randomPassword = generateRandomPassword(12);
console.log("Random Password:", randomPassword);
```

Output:

Random Password: Ydui8SUXFuE1

Question 5: Build a simple web page with a button. On clicking the button, change the background color of the page.

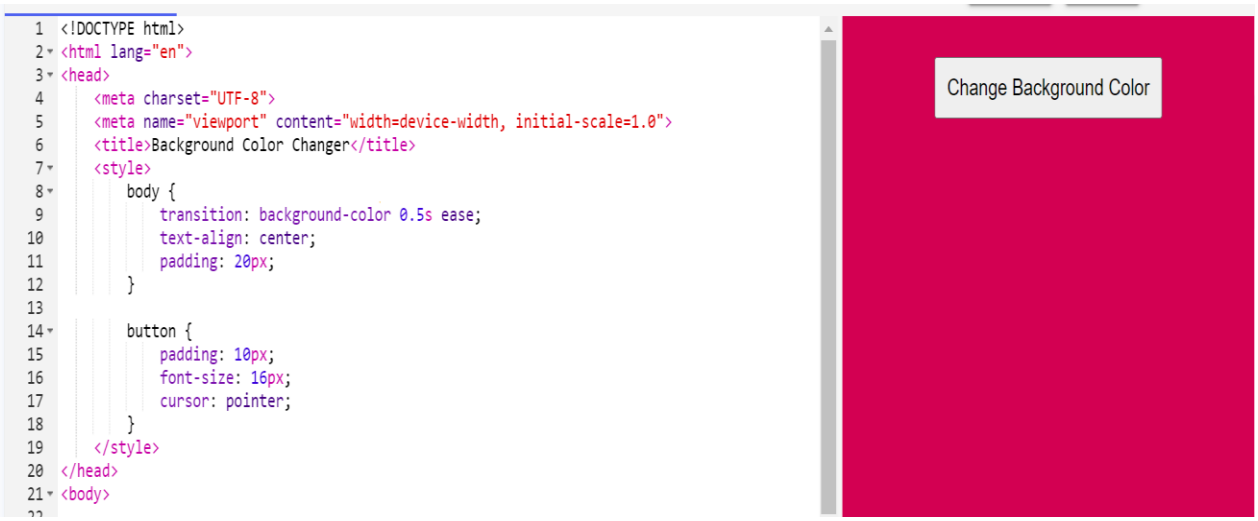
Coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Background Color Changer</title>
    <style>
        body {
            transition: background-color 0.5s ease;
            text-align: center;
            padding: 20px;
        }
```

```
        button
          {
            padding: 10px;
            font-size: 16px;
            cursor: pointer;
        }
    </style>
</head>
<body>
<button onclick="changeBackgroundColor()">Change Background Color</button>
<script>
    function changeBackgroundColor() {
        // Generate a random color in hex format
        const randomColor = '#' + Math.floor(Math.random()*16777215).toString(16);

        // Change the background color of the body
        document.body.style.backgroundColor = randomColor;
    }
</script>
</body>
</html>
```

Output:



Question 6: Create an image gallery using HTML and CSS. Implement JavaScript functionality to switch between images when clicking next or previous buttons.

Coding:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Gallery</title>
    <style>
        body
        {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 20px;
        }
        .gallery-container
        {
            max-width: 600px;
            margin: 0 auto;
            position: relative;
        }
        .gallery-image {
            max-width: 100%;
            height: auto;
            display: none;
        }
        .navigation-btn
        {
            cursor: pointer;
            font-size: 18px;
            padding: 10px;
            margin: 5px;
```

```
        }
    </style>
</head>
<body>
<div class="gallery-container">
    <img class="gallery-image" src="image1.jpg" alt="Image 1">
    <img class="gallery-image" src="image2.jpg" alt="Image 2">
    <img class="gallery-image" src="image3.jpg" alt="Image 3">
    <button class="navigation-btn" onclick="showPrevious()">Previous</button>
    <button class="navigation-btn" onclick="showNext()">Next</button>
</div>
<script>
    var currentIndex = 0;
    var images = document.querySelectorAll('.gallery-image');
    function showImage(index)
    {
        images.forEach(function(image)
        {
            image.style.display = 'none';
        });
        images[index].style.display = 'block';
        currentIndex = index;
    }
    function showNext()
    {
        currentIndex = (currentIndex + 1) % images.length;
        showImage(currentIndex);
    }
    function showPrevious()
    {
        currentIndex = (currentIndex - 1 + images.length) % images.length;
        showImage(currentIndex);
    }
```

```
    // Show the first image initially
    showImage(currentIndex);
</script>
</body>
</html>
```

Output:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Gallery</title>
    <style>
        body
        {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 20px;
        }

        .gallery-container
        {
            max-width: 600px;
            margin: 0 auto;
            position: relative;
        }
```

Image 1

Previous Next

Question 7: Write a JavaScript program using the Fetch API to fetch and display data from an external API (e.g., NASA's APOD - Astronomy Picture of the Day).

Coding:

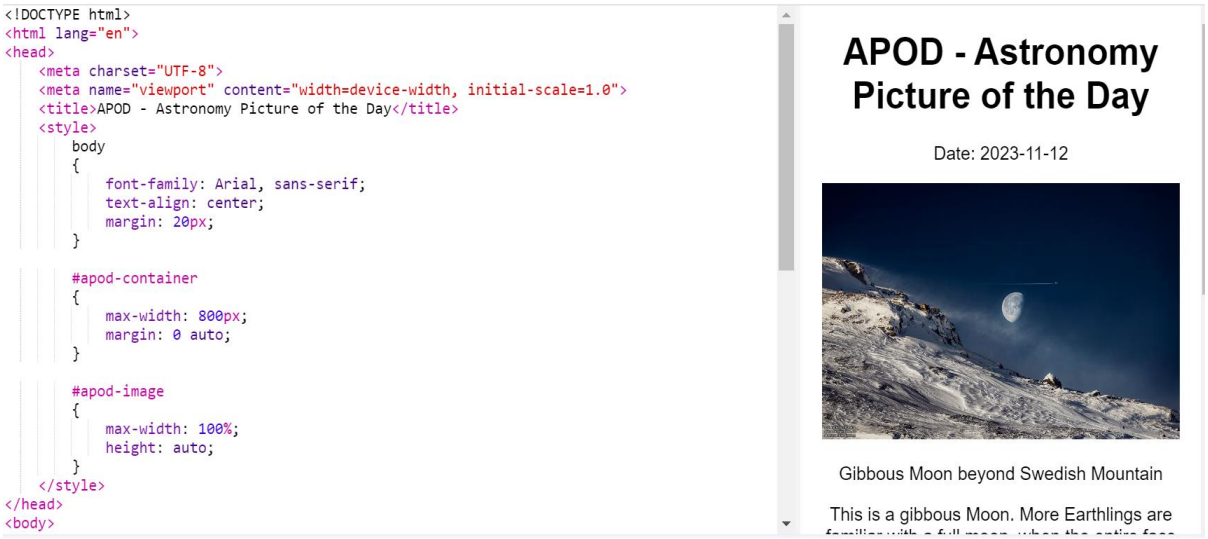
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>APOD - Astronomy Picture of the Day</title>
    <style>
        body
        {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 20px;
```

```
        }
        #apod-container
        {
            max-width: 800px;
            margin: 0 auto;
        }
        #apod-image
        {
            max-width: 100%;
            height: auto;
        }
    </style>
</head>
<body>
<div id="apod-container">
    <h1>APOD - Astronomy Picture of the Day</h1>
    <p id="apod-date"></p>
    <img id="apod-image" alt="Astronomy Picture of the Day">
    <p id="apod-title"></p>
    <p id="apod-explanation"></p>
</div>
<script>
    // NASA APOD API endpoint
    const apodUrl = 'https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY';

    // Fetch data from APOD API
    fetch(apodUrl)
        .then(response => response.json())
        .then(data =>
        {
            // Display APOD data
            document.getElementById('apod-date').innerText = 'Date: ' + data.date;
            document.getElementById('apod-image').src = data.url;
```

```
        document.getElementById('apod-title').innerText = data.title;
        document.getElementById('apod-explanation').innerText = data.explanation;
    })
    .catch(error => console.error('Error fetching APOD data:', error));
</script>
</body>
</html>
</html>
```

Output:



Question 8: Develop a program that simulates fetching data from two different APIs simultaneously using Promise. All and processes the results when both promises are resolved.

Coding:

```
function fetchDataFromApi1()
{
  return new Promise((resolve, reject) =>
  {
    // Simulate fetching data from API 1 (Replace with actual API endpoint)
    setTimeout(() => {
      const data = { resultFromApi1: "Data from API 1" };
      resolve(data);
```

```
        }, 2000); // Simulating a delay of 2 seconds
    });
}

function fetchDataFromApi2()
{
    return new Promise((resolve, reject) =>
    {
        // Simulate fetching data from API 2 (Replace with actual API endpoint)
        setTimeout(() => {
            const data = { resultFromApi2: "Data from API 2" };
            resolve(data);
        }, 1500); // Simulating a delay of 1.5 seconds
    });
}
// Simultaneously fetch data from both APIs using Promise.all()
Promise.all([fetchDataFromApi1(), fetchDataFromApi2()])
    .then(results => {
        const [resultFromApi1, resultFromApi2] = results;

        // Process the results when both promises are resolved
        console.log("Result from API 1:", resultFromApi1.resultFromApi1);
        console.log("Result from API 2:", resultFromApi2.resultFromApi2);
    })
    .catch(error => {
        // Handle errors if any of the promises is rejected
        console.error("Error fetching data:", error);
    });
```

Output:

Result from API 1: Data from API 1

Result from API 2: Data from API 2

Question 9: Implement a JavaScript class representing a 'Car'. Include properties like make, model, and year. Add methods to start the car and turn it off.

Coding:

```
class Car
{
   constructor(make, model, year)
{
      this.make = make;
      this.model = model;
      this.year = year;
      this.isEngineRunning = false;
   }

   start() {
      if (!this.isEngineRunning)
{
         console.log(`The ${this.year} ${this.make} ${this.model} is starting.`);
         this.isEngineRunning = true;
      }
else
{
         console.log(`The ${this.year} ${this.make} ${this.model} is already running.`);
      }
   }
   turnOff()
{
      if (this.isEngineRunning)
{
         console.log(`The ${this.year} ${this.make} ${this.model} is turning off.`);
         this.isEngineRunning = false;
      }
else
{
```

```
        console.log(`The ${this.year} ${this.make} ${this.model} is already turned off.`);
      }
   }
}
// Example usage
const myCar = new Car("Toyota", "Camry", 2022);
myCar.start(); // Output: The 2022 Toyota Camry is starting.
myCar.turnOff(); // Output: The 2022 Toyota Camry is turning off.
myCar.start(); // Output: The 2022 Toyota Camry is starting.
```

Output:

The 2022 Toyota Camry is starting.
The 2022 Toyota Camry is turning off.
The 2022 Toyota Camry is starting.

Question 10: Create an object representing a 'Library'. Implement methods to add a book, remove a book, and display the list of available books.

Coding:

```
const Library = {
   books: [],
   addBook: function(book)
{
      this.books.push(book);
      console.log(`${book.title} by ${book.author} has been added to the library.`);
   },
   removeBook: function(bookTitle)
{
      const index = this.books.findIndex(book => book.title === bookTitle);
      if (index !== -1)
{
         const removedBook = this.books.splice(index, 1)[0];
         console.log(`${removedBook.title} by ${removedBook.author} has been removed from the library.`);
```

```
        }
else
{
            console.log(`Book with title ${bookTitle} not found in the library.`);
        }
    },
    displayBooks: function()
{
        if (this.books.length === 0)
{
            console.log("The library is currently empty.");
        }
else
{
            console.log("List of available books in the library:");
            this.books.forEach(book =>
{
                console.log(`${book.title} by ${book.author}`);
            });
        }
    }
};
// Example usage
const book1 = { title: "The Great Gatsby", author: "F. Scott Fitzgerald" };
const book2 = { title: "To Kill a Mockingbird", author: "Harper Lee" };
const book3 = { title: "1984", author: "George Orwell" };
Library.addBook(book1);
Library.addBook(book2);
Library.displayBooks();
Library.removeBook("To Kill a Mockingbird");
Library.displayBooks();
```

Output:

The Great Gatsby by F. Scott Fitzgerald has been added to the library.

To Kill a Mockingbird by Harper Lee has been added to the library.
List of available books in the library:
The Great Gatsby by F. Scott Fitzgerald
To Kill a Mockingbird by Harper Lee
To Kill a Mockingbird by Harper Lee has been removed from the library.
List of available books in the library:
The Great Gatsby by F. Scott Fitzgerald