Experiment 2

Aim: To implement simple JavaScript Programs.

Theory:

1. Functions in JavaScript

What is a Function?

A function is a reusable block of code that performs a specific task. It can take inputs (called parameters), process them, and return an output. Functions help organize code, avoid repetition, and make programs easier to understand and maintain.

Example of a Function

```
function add(a, b) {
   return a + b;
}
const result = add(3, 5);
console.log(result);
```

Types of Functions

1. Named Functions:
   - Defined using the function keyword.
   - Can be called before or after their definition (due to hoisting).
2. Anonymous Functions:
   - Functions without a name.
   - Often used as callbacks or assigned to variables.

```
const greet = function(name) {
```

```
    return Hello, ${name}!;
  };
```

3. Arrow Functions (ES6):

- Shorter syntax for writing functions.
- Do not have their own this context.

```
const greet = (name) => Hello, ${name}!;
```

4. Immediately Invoked Function Expressions (IIFE):

- Functions that are executed immediately after they are defined.

```
(function() {
    console.log("This runs immediately!");
})();
```

2. Event Listeners in JavaScript

What is an Event Listener?

An event listener is a function that waits for a specific event (e.g., a click, keypress, or form submission) to occur on a webpage element. When the event occurs, the listener executes a callback function to handle the event.

Example of an Event Listener

```
const button = document.querySelector("button");

button.addEventListener("click", function() {
   alert("Button clicked!");
});
```

Common Event Types

- Mouse Events:
  - click: When an element is clicked.
  - mouseover: When the mouse pointer is moved over an element.
  - mouseout: When the mouse pointer leaves an element.

- Keyboard Events:
  - keydown: When a key is pressed down.
  - keyup: When a key is released.

- Form Events:
  - submit: When a form is submitted.
  - input: When the value of an input field changes.

- Window Events:
  - load: When the webpage finishes loading.
  - resize: When the browser window is resized.

Programs:

1. Digital Clock

Code:

Html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Digital Clock</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="clock">
    <span id="hours">00</span>:
    <span id="minutes">00</span>:
    <span id="seconds">00</span>
    <span id="ampm">AM</span>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

CSS:

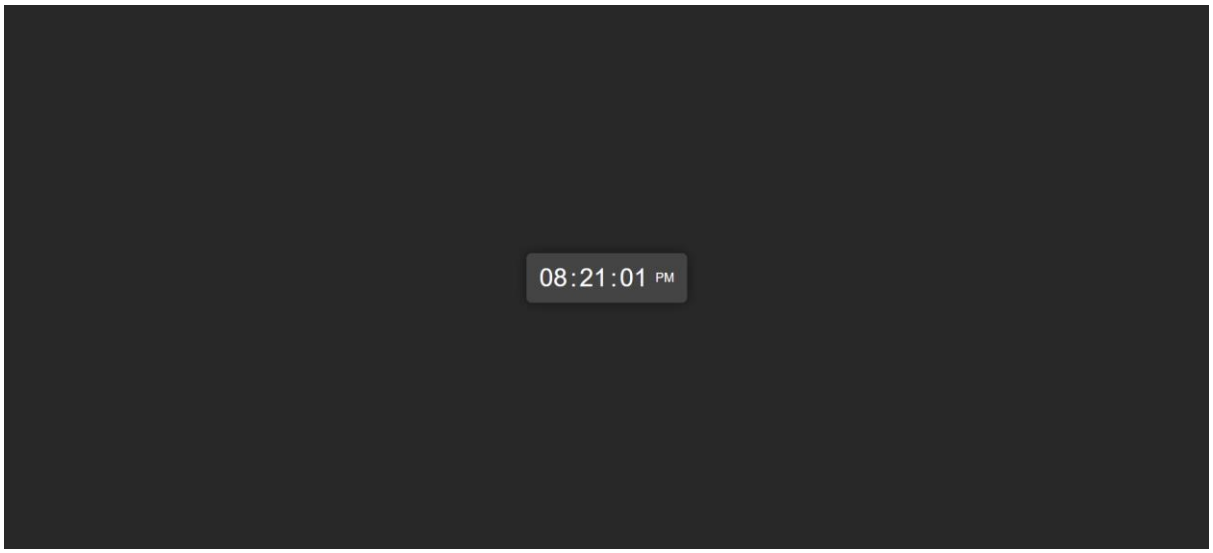
```
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  background-color: #282828;
  font-family: 'Arial', sans-serif;
}

.clock {
  display: flex;
  align-items: center;
  background-color: #444;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);
  color: #fff;
  font-size: 3rem;
}

.clock span {
  margin: 0 5px;
}

#ampm {
  font-size: 1.5rem;
  margin-left: 10px;
}
```

JavaScript:

```
function updateClock() {
  const now = new Date();
  const hours = now.getHours();
```

```
    const minutes = now.getMinutes();
    const seconds = now.getSeconds();
    const ampm = hours >= 12 ? 'PM' : 'AM';


    const formattedHours = hours % 12 || 12;


    const displayHours =
String(formattedHours).padStart(2, '0');
    const displayMinutes =
String(minutes).padStart(2, '0');
    const displaySeconds =
String(seconds).padStart(2, '0');
    document.getElementById('hours').textCont
ent = displayHours;
    document.getElementById('minutes').textC
ontent = displayMinutes;
    document.getElementById('seconds').textC
ontent = displaySeconds;
    document.getElementById('ampm').textCon
tent = ampm;
}


setInterval(updateClock, 1000);


updateClock();
```

Output:



2. Background Changer

Code:

Html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Background Color</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Background Color Changer</h1>
    <p>The background color changes every 5 seconds.</p>
    <script src="script.js"></script>
</body>
</html>
```

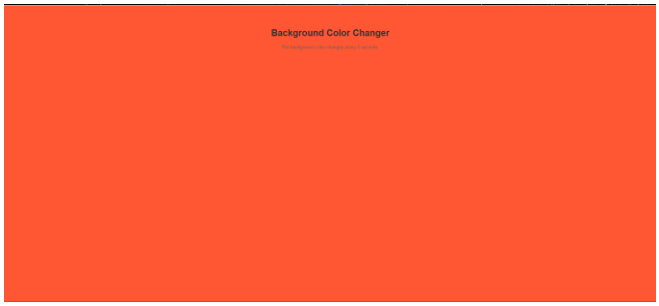
CSS:

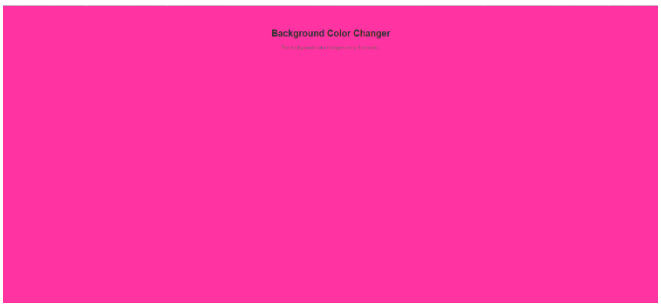
```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 50px;
    transition: background-color 1s ease;
}
h1 {
    color: #333;
}

p {
    color: #666;
}
```

JS:

```
const colors = ["#FF5733", "#33FF57", "#3357FF", "#FF33A1", "#33FFF5", "#F5FF33"];
let currentIndex = 0;
function changeBackgroundColor() {
    document.body.style.backgroundColor = colors[currentIndex];
    currentIndex = (currentIndex + 1) % colors.length;
}
setInterval(changeBackgroundColor, 5000);
changeBackgroundColor();
```

Output:





3. Simple Calculator

Code:

Html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="calc.css">
  <script src="calc.js" defer></script>
</head>
<body>
  <div>
    <form action="">
      <input type="text" placeholder="enter number 1" id="ip1">
      <input type="text" placeholder="enter number 2" id="ip2">
    </form>
  </div>
  <div class="down">
    <label for="">Answer: <span id="result"></span></label>
    <button id="addBtn" class="btn">Add</button>
    <button id="subBtn" class="btn">Subtract</button>
  </div>

</body>
</html>
```

CSS:

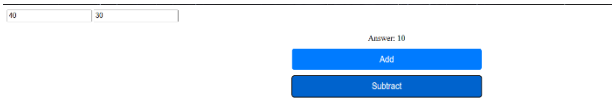
```
.down {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 10px;
  margin-top: 20px;
}

.btn {
  width: 25%;
```

```
    padding: 10px;
    font-size: 1rem;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #007bff;
    color: white;
    transition: background-color 0.3s ease;
}

.btn:hover {
    background-color: #0056b3;
}

#result {
    font-weight: bold;
    color: #2c3e50;
}
```

JS:

```
const input1 = document.getElementById('ip1');
const input2 = document.getElementById('ip2');
const addBtn = document.getElementById('addBtn');
const subBtn = document.getElementById('subBtn');
const resultSpan = document.getElementById('result');

function addNumbers() {
    const num1 = parseFloat(input1.value);
    const num2 = parseFloat(input2.value);

    if (isNaN(num1) || isNaN(num2)) {
        resultSpan.textContent = "Invalid input";
    } else {
        const sum = num1 + num2;
        resultSpan.textContent = sum;
    }
}

function subtractNumbers() {
    const num1 = parseFloat(input1.value);
    const num2 = parseFloat(input2.value);

    if (isNaN(num1) || isNaN(num2)) {
        resultSpan.textContent = "Invalid input";
    } else {
        const difference = num1 - num2;
        resultSpan.textContent = difference;
    }
}

addBtn.addEventListener('click', addNumbers);
subBtn.addEventListener('click', subtractNumbers);
```

Output:



4. Bulb

Code:

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="bulb.css">
    <script src="bulb.js" defer></script>
</head>
<body>
    <img src="on.png" alt="bulb_on" id="bulb">
    <button id="toggle">toggle</button>
</body>
</html>
```

CSS:

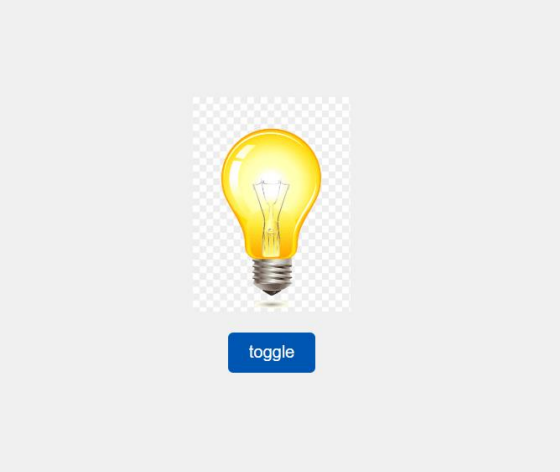
```
img{
    width: 100px;
    height: 100px;
}
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    margin: 0;
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}

#bulb {
    width: 150px;
    height: auto;
    margin-bottom: 20px;
}

#toggle {
    padding: 10px 20px;
    font-size: 1rem;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #007bff;
    color: white;
```

```
    transition: background-color 0.3s ease;
}

#toggle:hover {
    background-color: #0056b3;
}
```

JS:

```
const bulb = document.getElementById('bulb');
const toggleButton = document.getElementById('toggle');

toggleButton.addEventListener('click', function () {
    if (bulb.src.endsWith('on.png')) {
        bulb.src = 'off.jpg';
    } else {
        bulb.src = 'on.png';
    }
});
```

Output:



5. Paragraph hide/show on toggle

Code:

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Show/Hide Content</title>
    <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
    <button id="toggleButton">Hide Content</button>
    <p id="content">This is the content that will be hidden or shown when you click the button.</p>

    <script src="script.js"></script>
</body>
</html>
```

CSS:

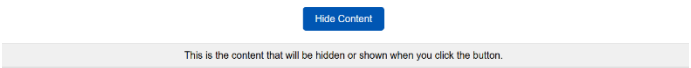
```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 20px;
}

#content {
    margin-top: 20px;
    padding: 10px;
    background-color: #f0f0f0;
    border: 1px solid #ccc;
    border-radius: 5px;
    transition: opacity 0.3s ease;
}

button {
    padding: 10px 20px;
    font-size: 1rem;
    cursor: pointer;
    border: none;
    border-radius: 5px;
```

```
    background-color: #007bff;
    color: white;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #0056b3;
}
```

JS:

```
const toggleButton = document.getElementById('toggleButton');
const content = document.getElementById('content');

toggleButton.addEventListener('click', function () {
    if (content.style.display === 'none') {
        content.style.display = 'block';
        toggleButton.innerText = 'Hide Content';
    } else {
        content.style.display = 'none';
        toggleButton.innerText = 'Show Content';
    }
});
```

Output:



6. Dynamic Portfolio

Code:

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Professional Portfolio</title>
  <link rel="stylesheet" href="styles.css">

  <!-- Internal CSS -->
  <style>
    .project-card {
      background: #f8f9fa;
      padding: 1rem;
      border-radius: 8px;
      margin-bottom: 1rem;
      transition: transform 0.2s;
    }

    .project-card:hover {
      transform: translateY(-5px);
      box-shadow: 0 5px 15px rgba(0,0,0,0.1);
    }

    .skill-item {
      background: #e9ecef;
      padding: 0.8rem;
      border-radius: 6px;
      text-align: center;
    }

    input, textarea {
      padding: 0.8rem;
      border: 1px solid #ddd;
      border-radius: 4px;
      width: 100%;
    }

    button {
      background: #6bd74d;
      color: black;
      padding: 0.8rem 1.5rem;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      transition: background 0.2s;
    }

    button:hover {
      background: #199104;
      color: white;
    }
  </style>
```

```
</head>

<body>

  <div class="container">

    <!-- Header with inline CSS for specific customization -->

    <header class="header" style="background: linear-gradient(to right, #f8f9fa, #e9ecef);">

      <img src="profile2.jpg" alt="Profile Picture" class="profile-img">

      <h1 style="color: #2c5048; margin-top: 1rem;">Rushil Rohra</h1>

      <p style="color: #666; font-size: 1.1rem;">Full-Stack Developer</p>

    </header>


    <section class="section">

      <h2 class="section-title">About Me</h2>

      <p>Passionate full-stack developer with expertise in building scalable web applications. Experienced in modern frameworks and cloud technologies. Committed to writing clean, efficient code and creating exceptional user experiences.</p>

    </section>


    <section class="section">

      <h2 class="section-title">Projects</h2>

      <div class="project-card">

        <h3>Resume Template</h3>

        <p>A resume template that is device responsive</p>

      </div>

      <div class="project-card">

        <h3>Currency Converter</h3>

        <p>Used basic JS concepts and API to make a simple currency converter</p>

      </div>

    </section>


    <section class="section">

      <h2 class="section-title">Skills</h2>

      <div class="skills-grid">

        <div class="skill-item">HTML</div>

        <div class="skill-item">CSS</div>

        <div class="skill-item">JAVA</div>

        <div class="skill-item">Python</div>

        <div class="skill-item">TypeScript</div>

        <div class="skill-item">ABC</div>

      </div>

    </section>


    <section class="section">

      <h2 class="section-title">Contact</h2>

      <form class="contact-form">

        <input type="text" placeholder="Name" required>

        <input type="email" placeholder="Email" required>

        <textarea placeholder="Message" rows="4" required></textarea>

        <button type="submit">Send Message</button>

      </form>

    </section>

  </div>
```

```
    <script src="script.js"></script>
</body>
</html>
```

CSS:

```
/* Reset and base styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

/* Layout and container styles */
body {
    background: linear-gradient(135deg, #40E0D0, #4169E1);
    min-height: 100vh;
    padding: 2rem;
    line-height: 1.6;
    color: #333;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background: white;
    border-radius: 15px;
    box-shadow: 0 10px 20px rgba(0,0,0,0.1);
    overflow: hidden;
}

/* Header styles */
.header {
    text-align: center;
    padding: 2rem;
    background: #f8f9fa;
}

.profile-img {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    object-fit: cover;
    border: 5px solid white;
    box-shadow: 0 5px 15px rgba(0,0,0,0.2);
    transition: transform 0.3s ease;
}

.profile-img:hover {
    transform: scale(1.05);
}

/* Section styles */
.section {
    padding: 2rem;
    border-bottom: 1px solid #eee;
}

.section:last-child {
    border-bottom: none;
}
```

```
.section-title {
    color: #2c3e50;
    margin-bottom: 1.5rem;
    font-size: 1.5rem;
    border-bottom: 2px solid #8df58d;
    display: inline-block;
    padding-bottom: 0.5rem;
}

/* Project styles */
.projects-container {
    display: grid;
    gap: 1.5rem;
    margin-top: 1rem;
}

/* Skills section */
.skills-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 1rem;
    margin-top: 1rem;
}

/* Contact form */
.contact-form {
    display: grid;
    gap: 1rem;
    max-width: 600px;
    margin: 0 auto;
}

/* Typography */
h1 {
    font-size: 2.5rem;
    margin-bottom: 0.5rem;
}

h2 {
    font-size: 2rem;
    margin-bottom: 1rem;
}

h3 {
    font-size: 1.5rem;
    margin-bottom: 0.5rem;
}

p {
    margin-bottom: 1rem;
    line-height: 1.6;
}

/* Links */
a {
    color: #94eea7;
    text-decoration: none;
    transition: color 0.2s ease;
}
```

```
a:hover {
  color: #2fe02f;
  text-decoration: underline;
}

/* Media Queries */
@media (max-width: 768px) {
  body {
    padding: 1rem;
  }

  .container {
    border-radius: 10px;
  }

  .section {
    padding: 1.5rem;
  }

  .skills-grid {
    grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  }

  h1 {
    font-size: 2rem;
  }

  h2 {
    font-size: 1.75rem;
  }

  h3 {
    font-size: 1.25rem;
  }
}

@media (max-width: 480px) {
  .profile-img {
    width: 120px;
    height: 120px;
  }

  .section {
    padding: 1rem;
  }

  .skills-grid {
    grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
  }
}

@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
```

```
}

.section {
  animation: fadeIn 0.8s ease-out forwards;
}

.section:nth-child(1) {
  animation-delay: 0.2s;
}

.section:nth-child(2) {
  animation-delay: 0.4s;
}

.section:nth-child(3) {
  animation-delay: 0.6s;
}

.section:nth-child(4) {
  animation-delay: 0.8s;
}

@keyframes cardHover {
  from {
    transform: scale(1);
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
  }
  to {
    transform: scale(1.05);
    box-shadow: 0 10px 20px rgba(0, 0, 0,
0.2);
  }
}

.project-card {
  transition: transform 0.3s ease, box-shadow
0.3s ease;
}

.project-card:hover {
  animation: cardHover 0.3s ease forwards;
}

@keyframes profileImageHover {
  from {
    transform: scale(1);
  }
  to {
    transform: scale(1.1);
  }
}

.profile-img {
  transition: transform 0.3s ease;
}

.profile-img:hover {
  animation: profileImageHover 0.3s ease
forwards;
}

.display-details {
  margin-top: 2rem;
```

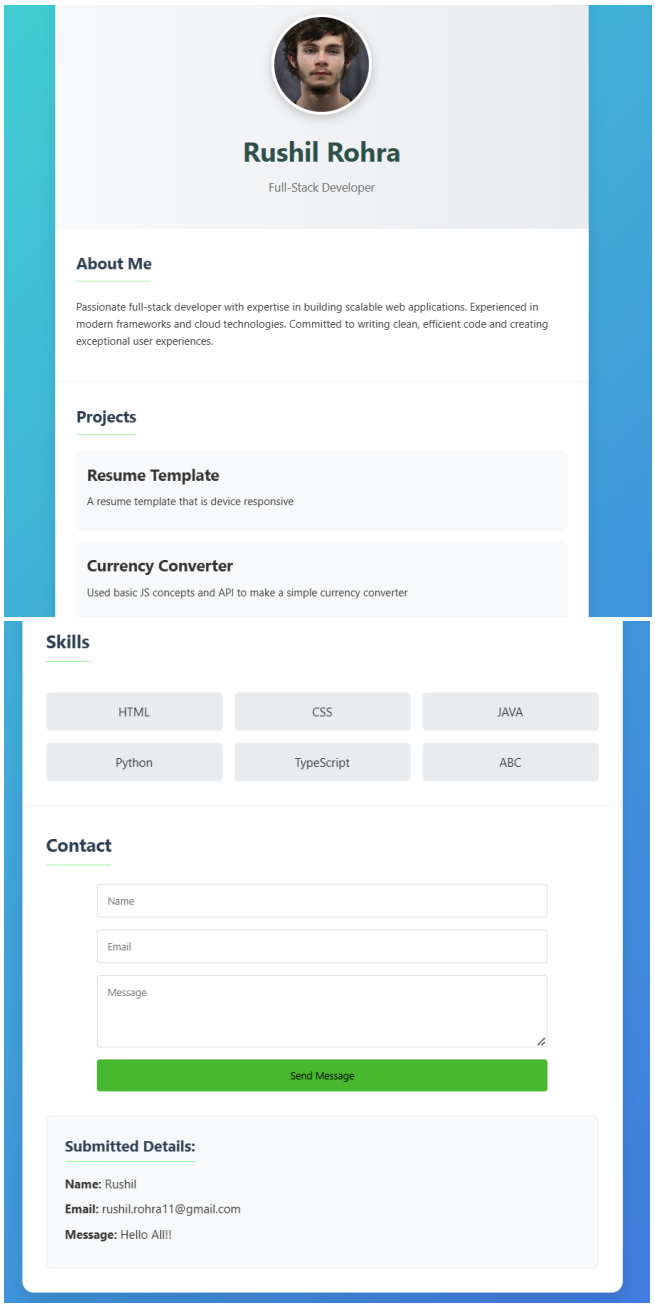
```
    padding: 1.5rem;
    background: #f8f9fa;
    border-radius: 8px;
    border: 1px solid #eee;
    animation: fadeIn 0.5s ease-out;
}

.display-details h3 {
    margin-bottom: 1rem;
    color: #2c3e50;
    font-size: 1.25rem;
    border-bottom: 2px solid #8df58d;
    display: inline-block;
    padding-bottom: 0.25rem;
}

.display-details p {
    margin-bottom: 0.5rem;
    color: #333;
}
/* Print styles */
@media print {
    body {
        background: none;
        padding: 0;
    }

    .container {
        box-shadow: none;
        border: 1px solid #ddd;
    }

    .profile-img {
        box-shadow: none;
    }
}
```

JS:

```
const contactForm = document.querySelector('.contact-form');
const displayDetails = document.createElement('div');
displayDetails.className = 'display-details';
contactForm.parentElement.appendChild(displayDetails);

contactForm.addEventListener('submit', function (event) {
    event.preventDefault();

    const name = contactForm.querySelector('input[type="text"]').value;
    const email = contactForm.querySelector('input[type="email"]').value;
    const message = contactForm.querySelector('textarea').value;

    const detailsHTML = `
        <h3>Submitted Details:</h3>
        <p><strong>Name:</strong> ${name}</p>
        <p><strong>Email:</strong> ${email}</p>
        <p><strong>Message:</strong> ${message}</p>
    `;
```

```
                                                    contactForm.reset();
    displayDetails.innerHTML = detailsHTML;                 });
```

Output:



Conclusion:

I was able to implement functions and action listeners effectively in JavaScript and implement dynamic elements on a static webpage. This helped the page to look more interactive and holds the user's attention.