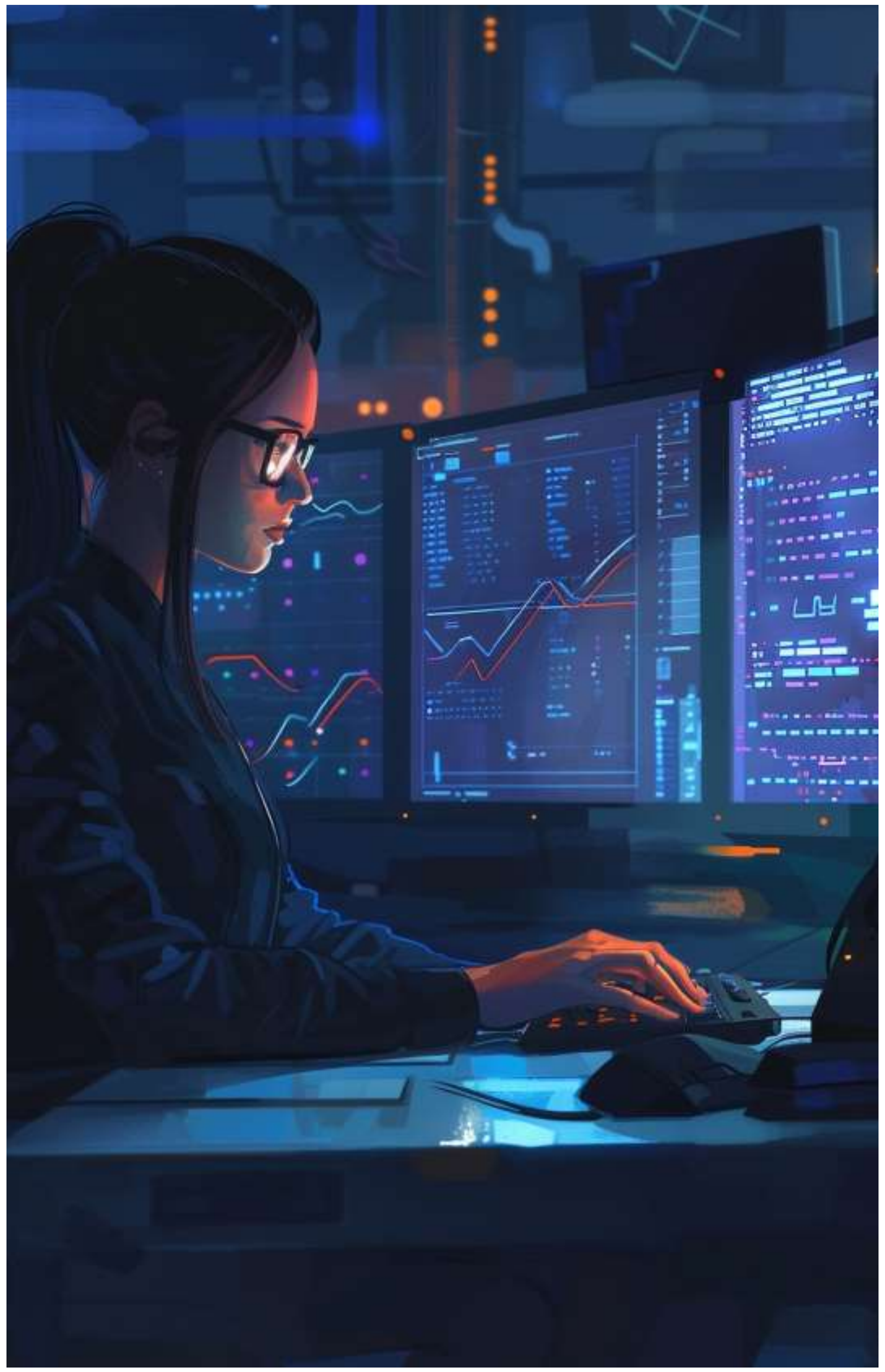
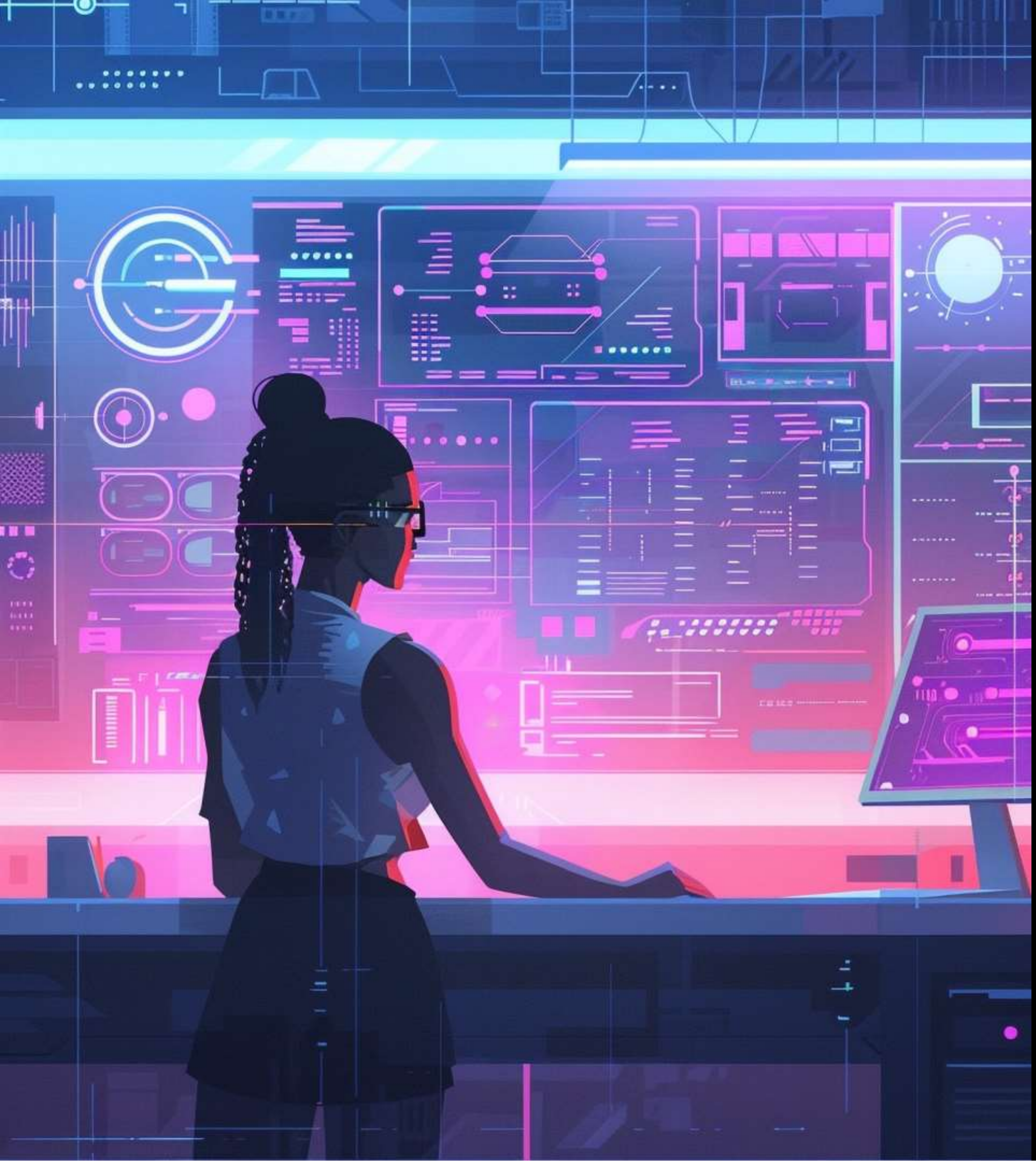


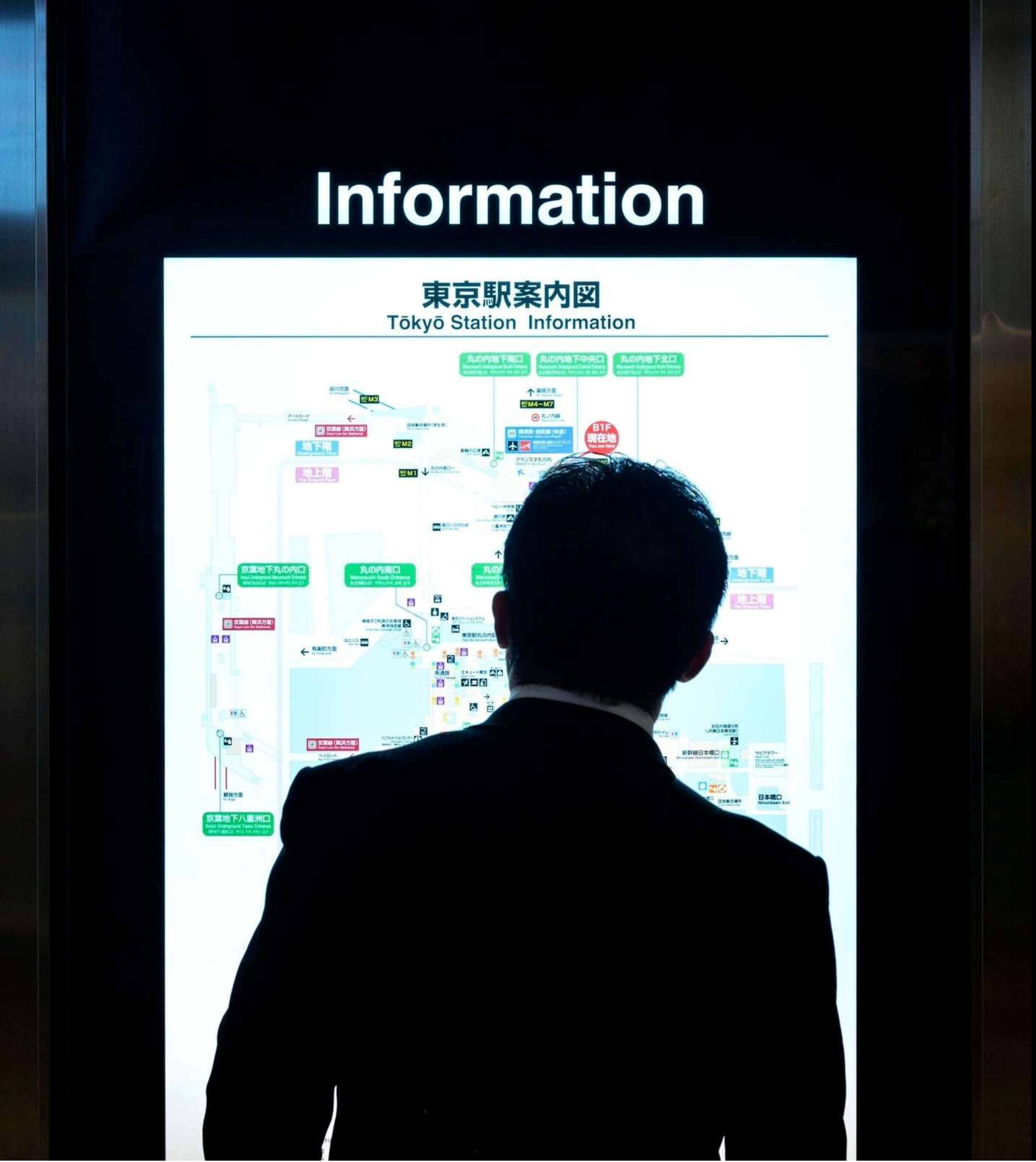
Linear Searching: Finding aNumber in Contact List

-Gowtham C (23isr016)
-Pranitha Senthilmurugan (23isr044)
-Sridhar S (23isr055)



Introduction to Linear Search:

Linear search is a fundamental algorithm used to find specific elements in a list. In this guide, we will explore its workings, advantages, and limitations. By the end, we will understand how to effectively implement linear search in our contact lists and other applications.



What is Linear Search?

Linear search is a straightforward algorithm that checks each element in a list sequentially until the desired element is found or the list ends. It is simple to implement and understand, making it a great choice for smaller datasets or unsorted lists.

How Linear Search Works

The process of Linear Search involves iterating through each element in the list. If the current element matches the target value, the search is successful. If not, the search continues until the end of the list is reached. This method is time-consuming for large datasets.

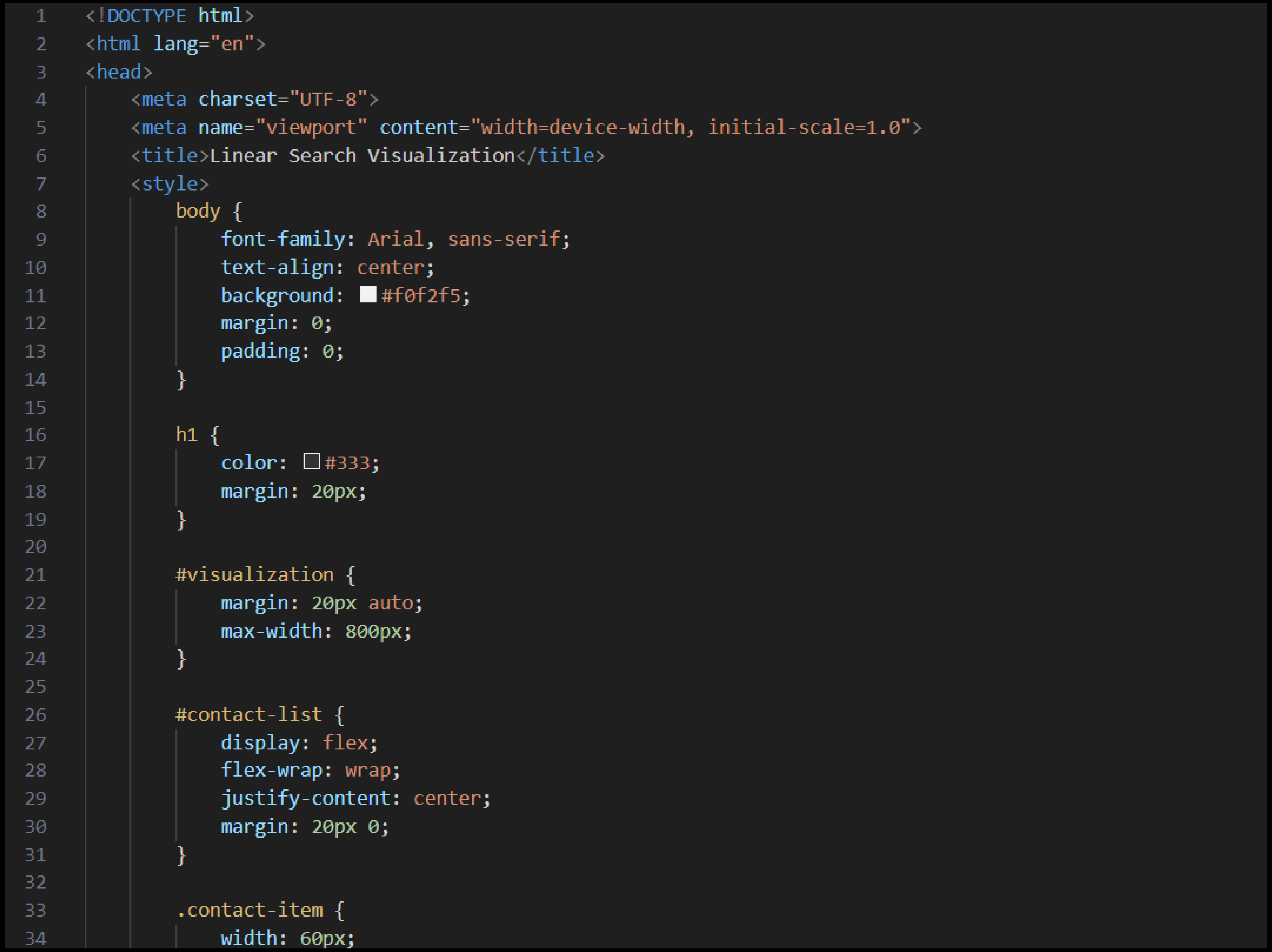
Advantages of Linear Search

One of the main advantages of Linear Search is its simplicity. It does not require the data to be sorted, making it versatile for various applications. Additionally, it is easy to implement in any programming language, which makes it accessible for beginners.

Limitations of Linear Search

Despite its advantages, linear search has notable limitations. Its complexity is O(n), making it inefficient for large datasets. This can lead to longer wait times when searching through extensive contact lists, prompting the need for more efficient algorithms in such cases.

Code



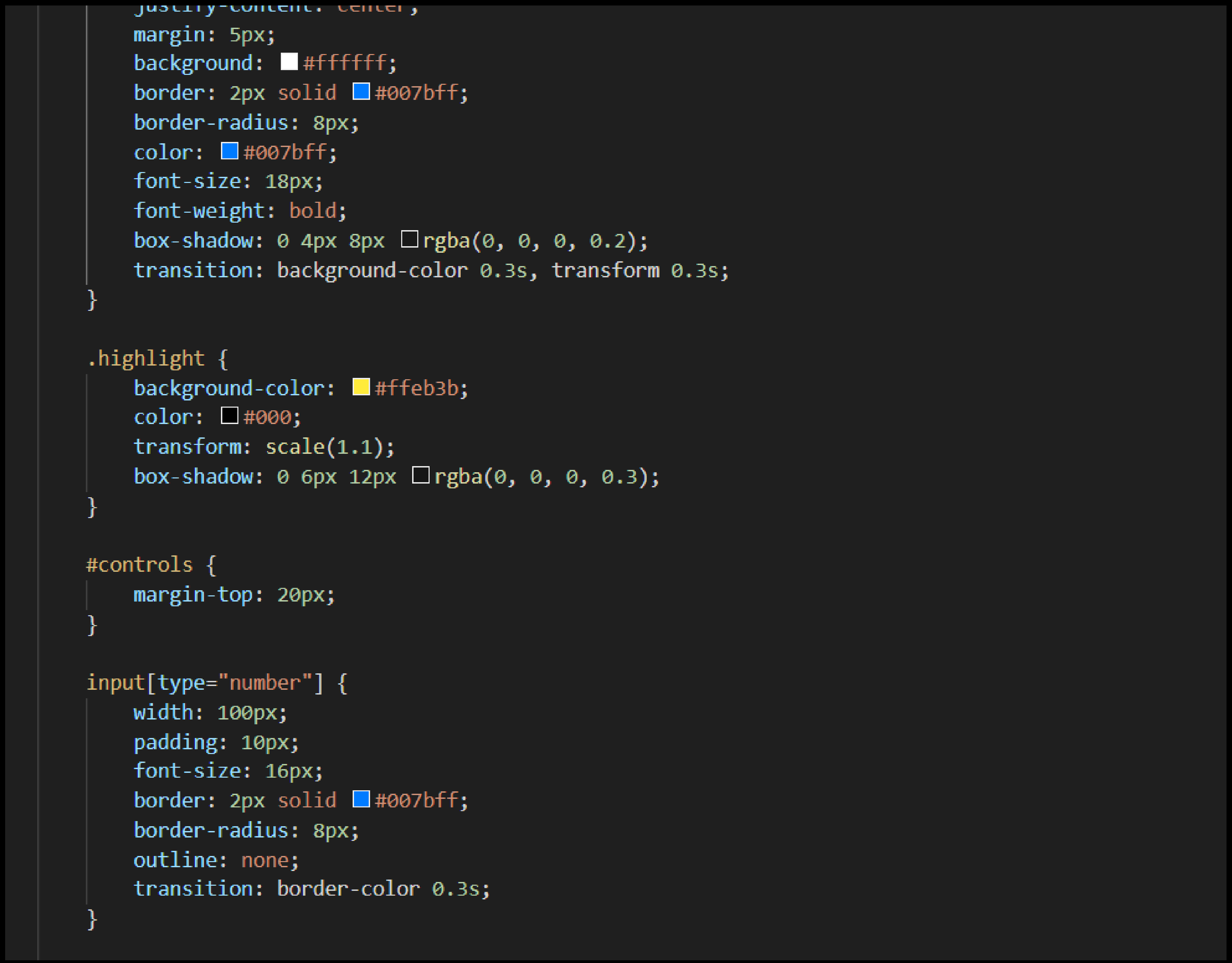
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Linear Search Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background: #f0f2f5;
            margin: 0;
            padding: 0;
        }

        h1 {
            color: #333;
            margin: 20px;
        }

        #visualization {
            margin: 20px auto;
            max-width: 800px;
        }

        #contact-list {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            margin: 20px 0;
        }

        .contact-item {
            width: 60px;
```

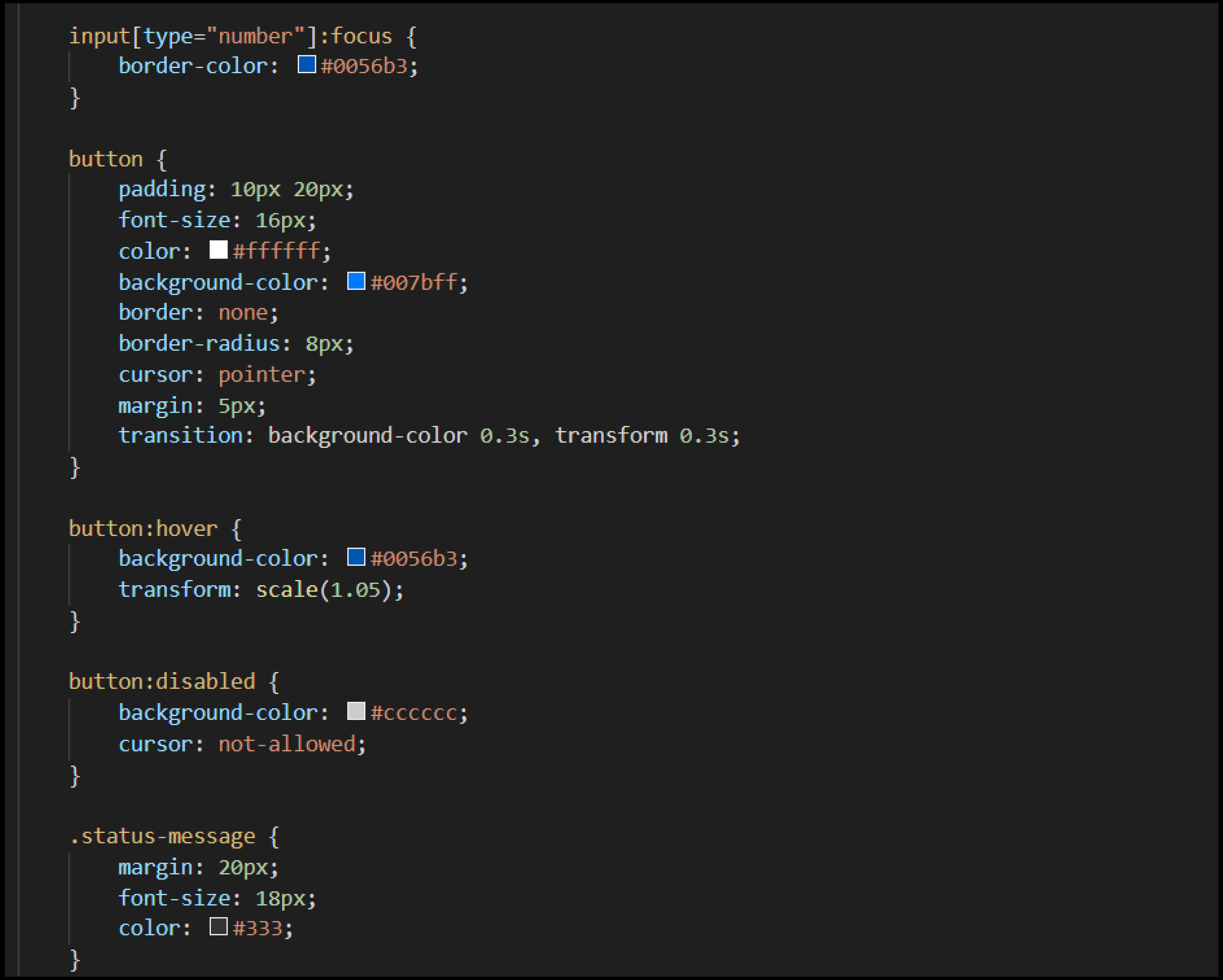


```css
            justify-content: center;
            margin: 5px;
            background: #ffffff;
            border: 2px solid #007bff;
            border-radius: 8px;
            color: #007bff;
            font-size: 18px;
            font-weight: bold;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            transition: background-color 0.3s, transform 0.3s;
        }

        .highlight {
            background-color: #ffeb3b;
            color: #000;
            transform: scale(1.1);
            box-shadow: 0 6px 12px rgba(0, 0, 0, 0.3);
        }

        #controls {
            margin-top: 20px;
        }

        input[type="number"] {
            width: 100px;
            padding: 10px;
            font-size: 16px;
            border: 2px solid #007bff;
            border-radius: 8px;
            outline: none;
            transition: border-color 0.3s;
        }
```



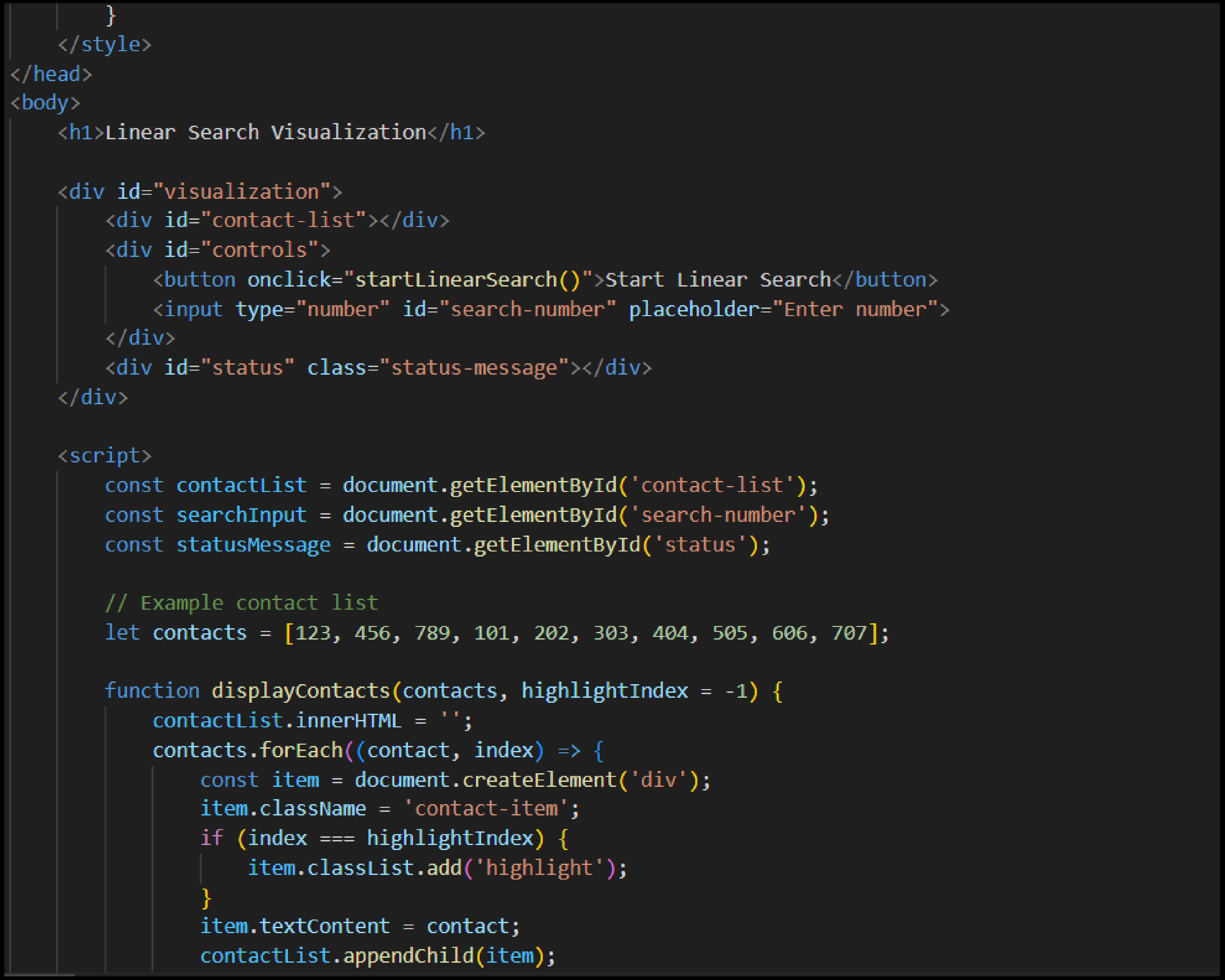
```css
input[type="number"]:focus {
    border-color: #0056b3;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    color: #ffffff;
    background-color: #007bff;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    margin: 5px;
    transition: background-color 0.3s, transform 0.3s;
}

button:hover {
    background-color: #0056b3;
    transform: scale(1.05);
}

button:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
}

.status-message {
    margin: 20px;
    font-size: 18px;
    color: #333;
}
```



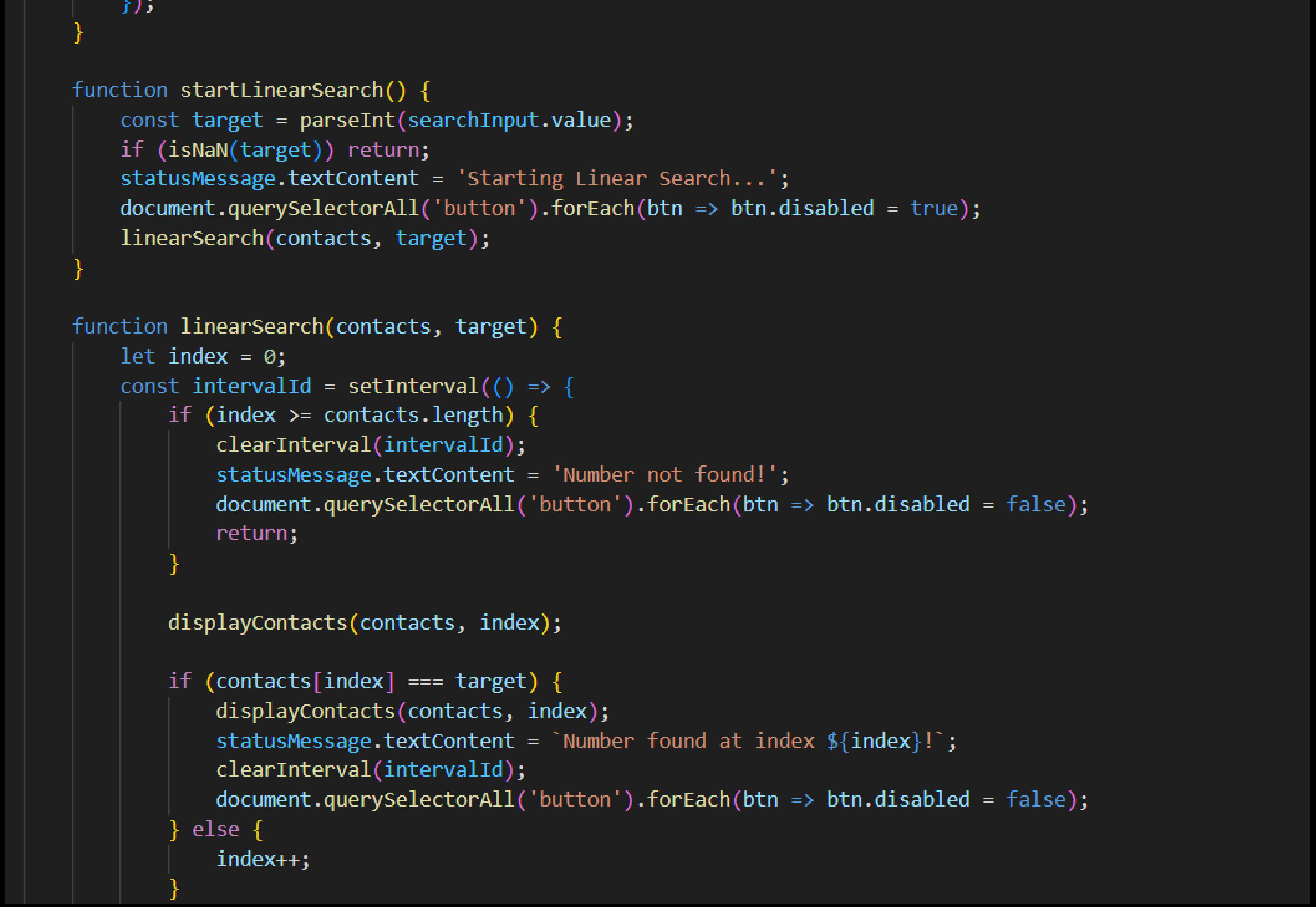
```html
        }
    </style>
</head>
<body>
    <h1>Linear Search Visualization</h1>

    <div id="visualization">
        <div id="contact-list"></div>
        <div id="controls">
            <button onclick="startLinearSearch()">Start Linear Search</button>
            <input type="number" id="search-number" placeholder="Enter number">
        </div>
        <div id="status" class="status-message"></div>
    </div>

    <script>
        const contactList = document.getElementById('contact-list');
        const searchInput = document.getElementById('search-number');
        const statusMessage = document.getElementById('status');

        // Example contact list
        let contacts = [123, 456, 789, 101, 202, 303, 404, 505, 606, 707];

        function displayContacts(contacts, highlightIndex = -1) {
            contactList.innerHTML = '';
            contacts.forEach((contact, index) => {
                const item = document.createElement('div');
                item.className = 'contact-item';
                if (index === highlightIndex) {
                    item.classList.add('highlight');
                }
                item.textContent = contact;
                contactList.appendChild(item);
```



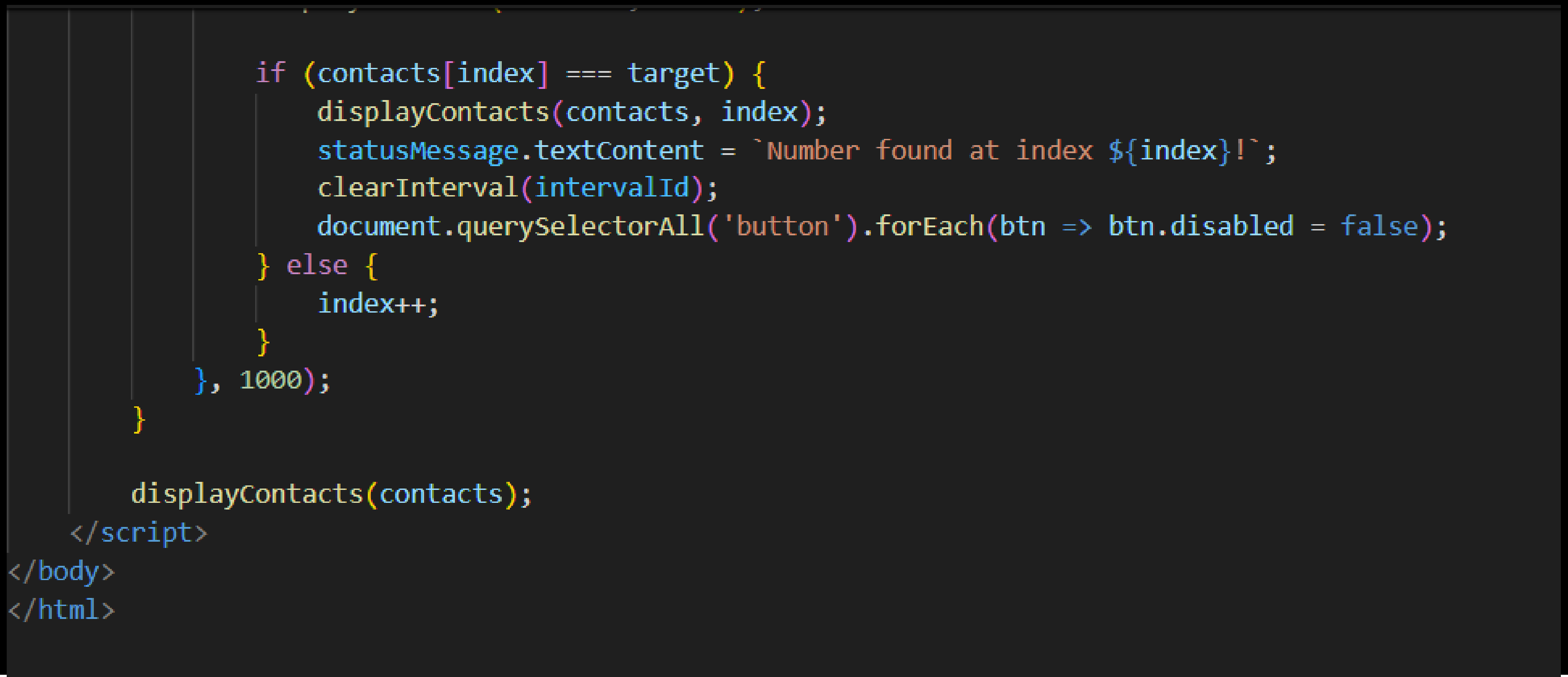
```
            });
        }

        function startLinearSearch() {
            const target = parseInt(searchInput.value);
            if (isNaN(target)) return;
            statusMessage.textContent = 'Starting Linear Search...';
            document.querySelectorAll('button').forEach(btn => btn.disabled = true);
            linearSearch(contacts, target);
        }

        function linearSearch(contacts, target) {
            let index = 0;
            const intervalId = setInterval(() => {
                if (index >= contacts.length) {
                    clearInterval(intervalId);
                    statusMessage.textContent = 'Number not found!';
                    document.querySelectorAll('button').forEach(btn => btn.disabled = false);
                    return;
                }

                displayContacts(contacts, index);

                if (contacts[index] === target) {
                    displayContacts(contacts, index);
                    statusMessage.textContent = `Number found at index ${index}!`;
                    clearInterval(intervalId);
                    document.querySelectorAll('button').forEach(btn => btn.disabled = false);
                } else {
                    index++;
                }
```



```

                if (contacts[index] === target) {
                    displayContacts(contacts, index);
                    statusMessage.textContent = `Number found at index ${index}!`;
                    clearInterval(intervalId);
                    document.querySelectorAll('button').forEach(btn => btn.disabled = false);
                } else {
                    index++;
                }
            }, 1000);
        }

        displayContacts(contacts);
    </script>
</body>
</html>
```

Introduction:
Linear search, also known as sequential search, is a simple algorithm used to find a specific element in a list by checking each element one by one until the desired element is found or the list ends.
Scenario:
Imagine you have a long list of contacts in your phone directory, and you need to find a specific contact’s phone number.

Steps:
Start at the Beginning: Begin with the first contact in the list.
Compare: Check if this contact is the one you’re looking for.
Move to the Next: If it’s not the correct contact, move to the next one in the list.
Repeat: Continue this process until you find the desired contact or reach the end of the list.

Issues in Linear Search

Definition: Sequentially checks each element in a list to find a target value.

Time Complexity:

Best Case: (O(1)) - Target is at the first position.

Worst Case: (O(n)) - Target is at the last position or not present.

Average Case: (O(n)) - On average, half the list is searched.

- **Efficiency Issues**:
  - **Scalability**: Performance degrades linearly with dataset size.
  - **Comparison**: Less efficient than binary search for large, sorted datasets.
- **Practical Limitations**:
  - **Large Datasets**: Inefficient due to high time complexity

Conclusion: Mastering Linear Search

In conclusion, Linear Search is a valuable tool for finding numbers in contact lists, especially in smaller datasets. Understanding its mechanics, advantages, and limitations equips you with the knowledge to decide when to use it effectively in your programming projects.

Thanks…