Write Code Effeciently

Advice for JavaScript Coder - 02

Hasibul Islam
@devhasibulsilam

Avoid Synchronous Operations: 01

```
// Bad: Synchronous operation
const result = expensiveSyncOperation();

// Good: Asynchronous operation
expensiveAsyncOperation((result) => {
  // Handle the result asynchronously
});
```

Use asynchronous APIs and functions provided by JavaScript and browser APIs, such as setTimeout, setInterval, and fetch. These APIs allow you to perform tasks asynchronously, preventing blocking operations.

Hasibul Islam
@devhasibulsilam

Avoid Synchronous Operations: 02

```javascript
// Using Promises
fetch(url)
  .then((response) => response.json())
  .then((data) => {
    // Handle the data asynchronously
  })
  .catch((error) => {
    // Handle errors
  }); // Using async/await
try {
  const response = await fetch(url);
  const data = await response.json(); // Handle the data asynchronously
} catch (error) {
  // Handle errors
}
```

Utilize Promises and async/await syntax for managing asynchronous operations. Promises provide a clean and structured way to handle asynchronous code, while async/await simplifies the syntax and improves readability.

Hasibul Islam
@devhasibulsilam

Optimize Event Handlers: 01

```
// Bad: Individual event handlers for each element
const buttons = document.querySelectorAll('.button');
buttons.forEach((button) => {
  button.addEventListener('click', (event) => {
    // Handle the click event
  });
});
// Good: Event delegation on a parent element
const parent = document.getElementById('parent');
parent.addEventListener('click', (event) => {
  if (event.target.classList.contains('button')) {
    // Handle the click event
  }
});
```

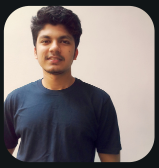
Implement event delegation, which involves attaching a single event handler to a parent element instead of multiple handlers to individual child elements. This technique reduces the number of event listeners and improves performance, especially for dynamically generated or large numbers of elements.javascriptCopycode

Hasibul Islam
@devhasibulsilam

Optimize Event Handlers: 02

```
// Throttling example
function throttle(callback, delay) {
  let lastExecutionTime = 0;
  return function () {
    const currentTime = Date.now();
    if (currentTime - lastExecutionTime > delay) {
      callback.apply(this, arguments);
      lastExecutionTime = currentTime;
    }
  };
}
window.addEventListener(
  'scroll',
  throttle(function () { // Handle the scroll event with throttling
    // write code here...
  }, 200)
);
```

Apply throttling or debouncing techniques to limit the frequency of executing event handlers, especially for events like scroll or resize that can trigger frequently. Throttling restricts the execution to a fixed interval, while debouncing postpones the execution until a certain period of inactivity

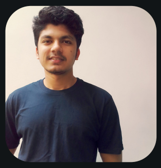

Hasibul Islam
@devhasibulsilam

Use Efficient Data Structures and Algorithms: 01

```javascript
// Good: Using an array for indexed access
const numbers = [1, 2, 3, 4, 5];
const thirdNumber = numbers[2];

// Good: Using a map for key-value pair lookups
const userMap = new Map();
userMap.set('John', { age: 30, email: 'john@example.com' });
const john = userMap.get('John');
```

Utilize appropriate data structures based on the requirements of your code. For example, use arrays for indexed access or stacks for last-in-first-out (LIFO) operations. Use objects or maps for key-value pair lookups or sets for storing unique values.

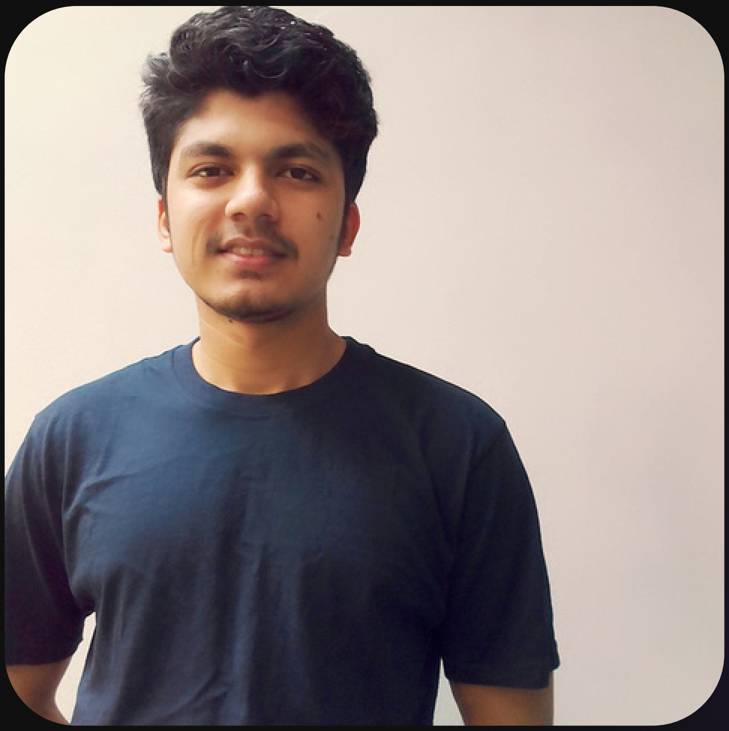


Hasibul Islam
@devhasibulsilam

Use Efficient Data Structures and Algorithms: 02

```
function binarySearch(array, target) {
  let start = 0;
  let end = array.length - 1;
while (start <= end) {
    const mid = Math.floor((start + end) / 2);
    if (array[mid] === target) {
      return mid;
    } else if (array[mid] < target) {
      start = mid + 1;
    } else { end = mid - 1; }
  }
  return -1;
}
```

Implement efficient algorithms for common tasks, such as searching, sorting, and manipulating data. For example, use binary search for sorted arrays, employ efficient sorting algorithms like quicksort or mergesort, and optimize data manipulation operations to minimize unnecessary iterations.



Hasibul Islam

@devhasibulislam

Did you find it useful?

Share your thoughts.

+8801906315901

devhasibulislam@gmail.com

https://devhasibulislam.vercel.app

Follow me on social

DM for Queries!

@devhasibulislam