Armen Melkumyan • 1st
Technical / Solutions Architect
1yr • Edited •

🚀 From JavaScript Technical Interviews to Real-World Challenges! 🌐

🛠️ The Challenge: Custom Event Emitter Implementation

An EventEmitter is a cornerstone of Node.js and browser-based JavaScript, facilitating asynchronous event handling with elegance and efficiency. The task? Create a lightweight EventEmitter from scratch, showcasing abilities in:

Subscribing to Events: Crafting a method to add event listeners seamlessly. 📫

Emitting Events: Implementing an emit function to trigger those listeners, passing along essential data. 🚀

Removing Event Listeners: Developing a strategy to detach listeners, ensuring our emitter is as dynamic as it is reliable. 🔧

Handling 'Once' Listeners: Adding a sprinkle of magic to handle listeners that vanish after a single use! ✨

📘 Step-by-Step Algorithm for Success

Initiate and Subscribe: Start by checking if our event already has listeners, and if not, create a space for them. It's all about making connections! 🤝

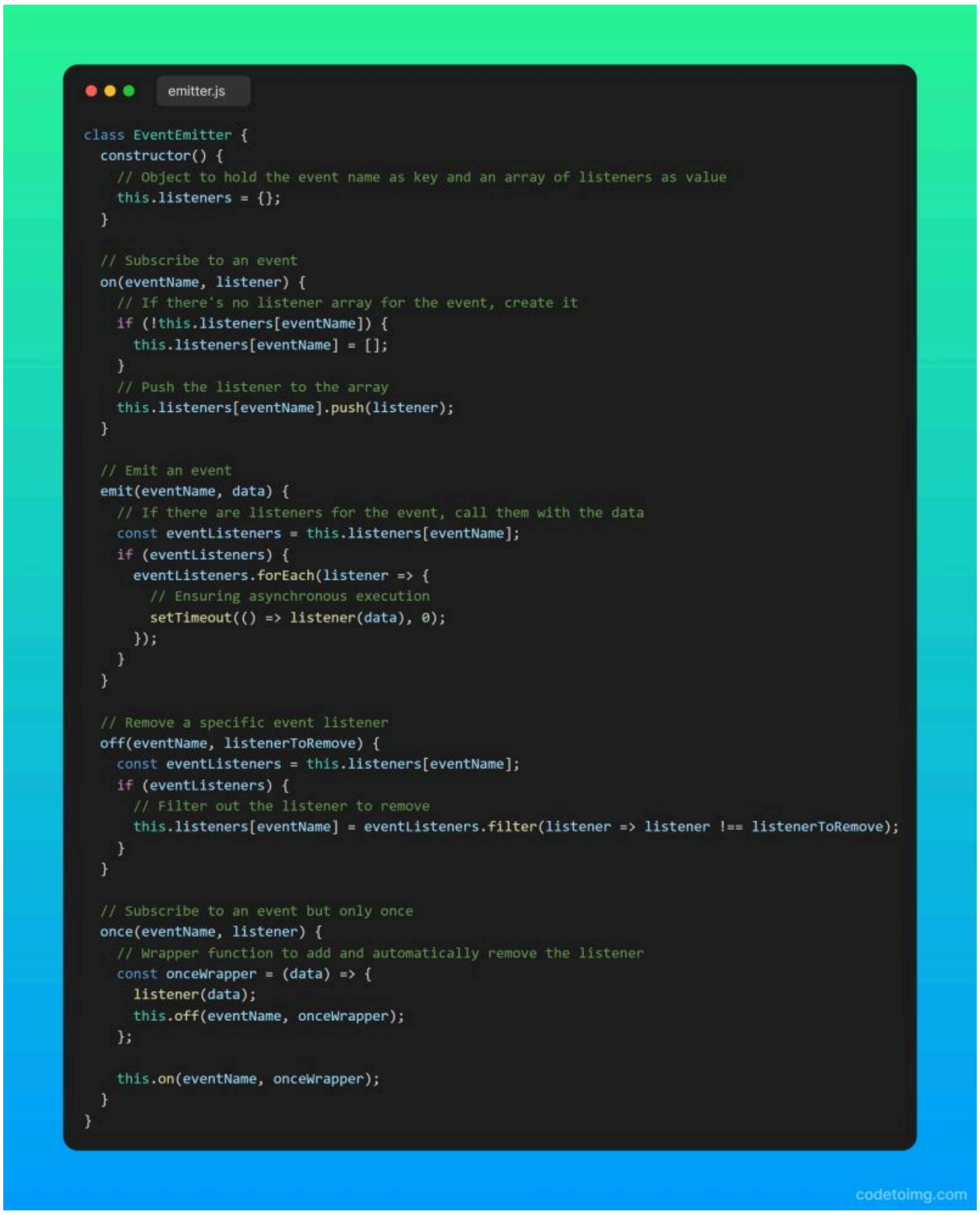
Broadcast Loud and Clear: Emit events with precision, ensuring each listener gets the memo — asynchronously, of course, to keep things smooth and non-blocking. 📢

Selective Hearing: Implement a method to remove specific listeners, because sometimes, it's okay to change your mind. 🙉

Once in a Lifetime: Design a mechanism for those once-off listeners, making each event count. 🎇

github link: https://lnkd.in/dw2baWHC

🏷️ #JavaScript #CodingChallenge #EventEmitter #TechInterviews #NodeJS #WebDevelopment #SoftwareEngineering #Programming #CodeNewbies #LearnToCode #DevCommunity



```javascript
class EventEmitter {
  constructor() {
    // Object to hold the event name as key and an array of listeners as value
    this.listeners = {};
  }

  // Subscribe to an event
  on(eventName, listener) {
    // If there's no listener array for the event, create it
    if (!this.listeners[eventName]) {
      this.listeners[eventName] = [];
    }
    // Push the listener to the array
    this.listeners[eventName].push(listener);
  }

  // Emit an event
  emit(eventName, data) {
    // If there are listeners for the event, call them with the data
    const eventListeners = this.listeners[eventName];
    if (eventListeners) {
      eventListeners.forEach(listener => {
        // Ensuring asynchronous execution
        setTimeout(() => listener(data), 0);
      });
    }
  }

  // Remove a specific event listener
  off(eventName, listenerToRemove) {
    const eventListeners = this.listeners[eventName];
    if (eventListeners) {
      // Filter out the listener to remove
      this.listeners[eventName] = eventListeners.filter(listener => listener !== listenerToRemove);
    }
  }

  // Subscribe to an event but only once
  once(eventName, listener) {
    // Wrapper function to add and automatically remove the listener
    const onceWrapper = (data) => {
      listener(data);
      this.off(eventName, onceWrapper);
    };

    this.on(eventName, onceWrapper);
  }
}
```

35 2 reposts