The built-
in `Array.sort()` method in JavaScript lets you sort your items much more succinctly and is generally more efficient than a custom merge sort

How to Sort with Built-in `sort()`

To sort your `items` array in descending order based on the `ratio` property, you can use:

Javascript Copy

```javascript
items.sort((a, b) => b.ratio - a.ratio);
```

This comparator subtracts `a.ratio` from `b.ratio`, so if `b.ratio` is greater than `a.ratio` (i.e., `b` should come before `a`), it returns a posit

Comparison: Built-in `sort()` vs. Custom Merge Sort

1. **Efficiency:**
   - **Built-
     in** `sort()`: Modern JavaScript engines (like V8 in Node.js) implement `sort()` in native code (often a variation of Timsort), which is hig
     level code, it tends to be faster than a custom merge sort implemented in JavaScript.
   - **Custom Merge Sort**: While a correctly implemented merge sort also offers O(n log n) performance in theory, a JavaScript version runs i
2. **Simplicity & Maintainability:**
   - **Built-in** `sort()`: Requires only a one-line callback and is a standard method that most developers are familiar with.
   - **Custom Merge Sort:** Involves writing multiple functions (like `mergeCall` and `merge`) and manually managing array indices, resulting i
3. **Reliability:**
   - **Built-in** `sort()`: Since it's part of the core language and has been battle-tested, you get a reliable and predictable behavior.
   - **Custom Merge Sort:** Your custom solution might yield the same result, but it could be less optimized or have edge cases that haven't b

Final Recommendation

Using the built-in `sort()` method is generally preferred for most tasks because it:

- **Is more succinct:** One-line sorting versus several lines of custom code.
- **Offers equivalent theoretical performance:** O(n log n) for both, but with optimizations in native code.
- **Is easier to read and maintain:** Most developers will immediately understand `items.sort((a, b) => b.ratio - a.ratio)`.

Example in Context:

Javascript Copy

```javascript
// Assume items is an array of items with a 'ratio' property.
console.log("Unsorted items:", items);

// Sort items in descending order by ratio using built-in sort
items.sort((a, b) => b.ratio - a.ratio);

console.log("Sorted items:", items);
```

This approach is clearer, more efficient, and easier to work with than implementing your own merge sort.