In JavaScript, arrays and objects (often used as hash tables or dictionaries) are fundamental structures for storing and manipulating data. However, they serve different purposes and have varying performance characteristics depending on the operations being performed. Here's a detailed comparison of their performance aspects:

**Arrays**

Arrays in JavaScript are ordered collections of values, and they are best for scenarios with contiguous integer-indexed use where order matters. Operations related to arrays include accessing elements by index, iterating, and manipulating the ends of the array.

* **Access**: Accessing an element by its index in an array is very fast—O(1), or constant time, because the position is directly computable from the index and base address of the array.
* **Insertions/Deletions**: At the beginning or middle of the array are costly (O(n)) because elements need to be shifted to maintain order. Append operations (push/pop at the end of the array) are typically efficient.
* **Search**: Searching for an element in an array without knowing the index takes O(n) since, in the worst case, every element needs to be checked.
* **Iteration**: Arrays are optimized for iterative operations and can be efficiently traversed using indexes.

**Objects as Hash Tables**

Objects in JavaScript are key-value stores where keys are strings (or symbols), and are typically used when associating unique keys with values. They provide efficient lookup, addition, and deletion operations.

* **Access**: Accessing data in a hash table (object) based on keys is very fast and generally considered to be O(1) on average—constant time complexity—owing to the direct computation of data location through the hash function.
* **Insertions/Deletions**: Adding or removing key-value pairs from an object also averages to O(1). There's no need to shift elements around, unlike arrays.
* **Search**: Searching for a value directly by key is O(1). If a search based on the value is required, then it necessitates O(n) since all pairs may need examination.

**Performance Comparison**

* **Insertion, Deletion, Access by Key/Index**: Both structures offer O(1) for these operations if keys or indexes are known. Arrays can be slower for insertions and deletions not at the end of the structure due to the shifting of elements.
* **Search**: Objects generally offer no advantage if the search for values (not keys) is needed, as both structures demand O(n) in such cases.
* **Memory Overhead**: Objects typically have more overhead than arrays due to storage of keys and the underlying hash table mechanism which may have additional space complexity.
* **Ordering**: Arrays inherently maintain the order of elements, offering advantages when this feature is required. Objects do not maintain any order (although as of ES2015, object properties retain their creation order, with certain exceptions like integer keys).

**Use Case Based Decision**

Choose the data structure based on actual needs:

* **Use arrays** when order is important, or you consistently operate at the ends of the data structure (efficient pushes and pops).
* **Use hash tables (objects or Maps)** when you need fast access, insertion, and removal by key without care for the order of elements.

Given JavaScript's dynamic nature, the actual implementation details and performance can vary between different JavaScript engines (like V8 for Chrome, SpiderMonkey for Firefox), but the big-O computational complexities generally hold true across environments. For truly intensive performance needs, profiling and benchmarking in the target JavaScript environment is recommended.