## Callback Functions

**Q: What is the purpose of callback functions in JavaScript?**  
A: Callback functions are used to handle asynchronous operations. They are functions passed as arguments to other functions and are invoked after certain tasks are completed. This allows for non-blocking code execution, enabling operations like network requests, timers, and event handling to be managed efficiently.

**Q: When should I use callback functions?**  
A: You should use callback functions when you need to execute a function after another function has completed its task. Common scenarios include handling asynchronous operations (like API calls or reading files), responding to events (like user interactions), and executing functions in sequence.

**Q: What’s the difference between regular functions and callback functions?**  
A: Regular functions are defined and invoked directly in your code, whereas callback functions are passed as arguments to other functions and are invoked after a certain event or operation has occurred. The main difference lies in how and when they are executed.

**Q: How common are callback functions vs regular functions?**  
A: Both callback functions and regular functions are commonly used in JavaScript. Regular functions are used for typical procedural code, while callback functions are crucial for handling asynchronous operations and events. Callback functions are particularly common in Node.js and front-end development with JavaScript.

**Q: What’s the difference between using a callback function and an event listener?**  
A: A callback function is passed to another function to be executed after the completion of a task, whereas an event listener is specifically used to handle events triggered by user interactions or other actions. Event listeners use callback functions to define the behavior that should occur when an event is fired.

**Q: Are callback functions only called as parameters in other functions?**  
A: While callback functions are primarily known for being passed as arguments to other functions, they can also be used in other ways, such as event handlers, within higher-order functions, and as custom iterators. However, their defining characteristic is that they are designed to be called at a later time after some action or event has occurred.

## Javascript Arrays

Q: What are array methods in javascript?

A: Array methods in JavaScript are built-in functions that can be applied to arrays to perform various operations. Some common array methods include push(), pop(), shift(), unshift(), slice(), splice(), concat(), forEach(), map(), filter(), reduce(), find(), findIndex(), indexOf(), includes(), join(), reverse(), and sort().

Q: Do all of these apply to lists?

A: In JavaScript, the term "lists" often refers to arrays, so all the array methods mentioned earlier apply to arrays.

Q: What's a simple example of using an array method?

A: Here's a simple example of using the map() method to double the elements of an array:

javascriptCopy codeconst numbers = [1, 2, 3, 4, 5];

const doubledNumbers = numbers.map(function(number) {

return number \* 2;

});

console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]

Q: What are linked lists?

A: A linked list is a linear data structure in which elements (called nodes) are not stored in contiguous memory locations. Instead, each node contains two parts: data and a reference (or pointer) to the next node. There are different types of linked lists, such as singly linked lists, doubly linked lists, and circular linked lists.

Q: So it creates a list of items that are in different areas, instead of just physically being in the same list?

A: Yes, that's correct. In a linked list, the elements (or nodes) are not stored in contiguous memory locations. Instead, each element points to the next element in the sequence.

Q: How could you easily see what's in a linked list?

A: To easily see what's in a linked list, you can create a method that traverses the list and collects the values of each node, then prints or returns these values. For example, you can add a method called toArray() that converts the linked list into an array for easy visualization.

Q: When would array methods be the most practical to use?

A: Array methods are particularly practical to use when dealing with data transformation, data processing, iterating through collections, and functional programming. Some common scenarios where array methods are useful include filtering data, transforming data, reducing data, finding elements, checking conditions, sorting data, merging arrays, and adding/removing elements.

Q: Are there less common array methods that weren't listed earlier?

A: Yes, there are several less common but useful array methods in JavaScript, such as flat(), flatMap(), from(), fill(), copyWithin(), entries(), keys(), values(), includes(), and findIndex().

Q: What is a subarray?

A: A subarray is a contiguous section of an array, consisting of elements that are part of the original array, taken in a sequence without skipping any elements. For example, [1, 2] and [2, 3, 4] are subarrays of the array [1, 2, 3, 4, 5]. Subarrays maintain the order of elements as they appear in the original array.