Here are several interview questions related to callback functions in Node.js:

1. \*\*What is a callback function in Node.js?\*\*

- A callback function is a function passed as an argument to another function, which is then invoked inside the outer function to complete some kind of action or operation.

2. \*\*Explain the purpose of using callback functions in Node.js.\*\*

- Callback functions are commonly used in Node.js for handling asynchronous operations, such as reading files, making network requests, or querying databases. They allow us to execute code after a particular task has completed without blocking the event loop.

8. \*\*Can you provide an example of using a callback function in Node.js?\*\*

- This question may require the candidate to provide code snippets demonstrating how callback functions are used in various scenarios, such as reading a file, making an HTTP request, or performing database queries.

These questions can help gauge a candidate's understanding of asynchronous programming concepts and their proficiency in using callback functions effectively in Node.js applications.

**Explain the concept & example of callback functions in JavaScript.**

**What are some methods to handle asynchronous code in JavaScript before Promises were introduced?**

1. **What is event delegation in JavaScript? Why is it useful?**
2. **What is the difference between == and === in JavaScript?**

In JavaScript, `==` and `===` are both comparison operators used to compare values. However, they behave differently:

1. `==` (Equality Operator):

- The equality operator compares the values on both sides for equality after converting them to a common type.

- If the values have different data types, JavaScript will attempt to convert them to a common type before making the comparison.

- This can sometimes lead to unexpected results due to type coercion.

2. `===` (Strict Equality Operator):

- The strict equality operator compares both the values and the data types of the operands.

- It returns true only if both the value and the data type are the same without any type conversion.

- Using `===` is generally considered safer and more predictable compared to `==` because it doesn't perform type coercion.

Here are a couple of examples to illustrate the difference:

```javascript

0 == false // true, because false is coerced to 0

0 === false // false, because the data types are different

1 == "1" // true, because "1" is coerced to 1

1 === "1" // false, because the data types are different

```

In summary, `==` checks for equality after type coercion, while `===` checks for strict equality without type coercion. It's often recommended to use `===` for most comparisons to avoid unexpected behavior caused by type coercion.

1. **What are the different ways to create objects in JavaScript?**
2. **Explain the concept of prototypal inheritance in JavaScript.**
3. **Monolithic vs microservices**
4. **Clustering** in Node.js refers to the ability to spawn multiple instances of a Node.js process to handle incoming requests. This is achieved using the built-in **cluster** module, which allows you to create child processes (workers) that share the same server port.