ANSWER 5-

Callbacks are functions that are passed as arguments to other functions and are executed at a later point in time. They are a common technique used in JavaScript to handle asynchronous operations. When an asynchronous operation completes, instead of waiting for the result, the program executes the callback function to handle the result.

In simple words, callbacks can be thought of as "function pointers" that allow you to specify what should happen after an asynchronous operation is finished. They enable you to define the behavior that should occur once the operation completes successfully or encounters an error.

However, if you have multiple nested asynchronous operations that depend on each other, it can lead to a situation called "callback hell." Callback hell occurs when there are multiple levels of nested callbacks, making the code structure complex and difficult to read and maintain. The code becomes deeply indented, making it hard to follow the flow and understand the logic. Here's an example of callback hell:

```
doSomething(function(result1) {
  doSomethingElse(result1, function(result2) {
    doSomethingElseAgain(result2, function(result3) {
      // ... and so on
    });
  });
});
```

As you can see, each asynchronous operation is nested within the callback of the previous operation. This nesting can become unwieldy and make the code harder to understand. To mitigate callback hell, modern JavaScript introduced techniques like Promises and

async/await, which provide more structured and readable ways to handle asynchronous operations