ADVANTAGES OF PREMISES OVER CALLBACKS

Promises offer several advantages over traditional callback methods when it comes to handling asynchronous tasks:

1. Improved Readability: Promises provide a more readable and straightforward syntax compared to nested callback functions. Promise chaining allows you to write asynchronous code in a more linear and sequential manner, making it easier to understand and maintain.
2. Error Handling: Promises have built-in error handling capabilities. You can attach a **.catch()** method at the end of a promise chain to handle any errors that occur during the asynchronous operation. This eliminates the need for explicit error handling in every callback function.
3. Callback Hell Avoidance: Promises help avoid the problem of "callback hell" or "pyramid of doom" where multiple nested callbacks make code difficult to read and reason about. With promises, you can chain asynchronous operations together using **.then()** and **.catch()** methods, resulting in cleaner and more manageable code.
4. Synchronous-Like Flow: Promises provide a more synchronous-like flow for asynchronous operations. With promises, you can use **async/await** syntax, which allows you to write asynchronous code that looks and behaves like synchronous code. This makes it easier to understand the control flow and write code in a more intuitive manner.
5. Composition and Error Propagation: Promises can be easily composed and combined. You can chain multiple promises together using **.then()** and handle errors at any point in the chain using **.catch()**. This allows for better error propagation and centralized error handling.
6. Promise Status and State: Promises have three different states: pending, fulfilled, and rejected. This allows you to determine the status of an asynchronous operation and handle it accordingly. Additionally, promises provide methods like **.all()** and **.race()** to handle multiple promises simultaneously.

Overall, promises provide a more elegant and efficient way to handle asynchronous tasks compared to traditional callback methods. They enhance code readability, error handling, and flow control, leading to more maintainable and reliable code.