**What are the differences between var, let, and const?**

**Var, let and const are very similar in the fact that they are all actions that allow us to declare a variable or assign a value to.The main differences between var let and const is the scope of the variable.**

**Let has a more local scope in the area of a function and will not exist outside the function. let also is known to be block scoped meaning that it is contained by {}. Let can be updated but not be declared differently. Let is useful for declaring a value inside of a function and having it exist only inside that function, and for the ability to reuse the same declaration name outside of the function.**

**Const is an assigned value  that is unchangeable and is a constant throughout the entire code file.Const has a global scope in that it will remain unchanged regardless if the value is assigned differently in separate functions.It is also good practice to use all caps when declaring a constant.**

**var was the original variable expression before ES6 and can be used globally or locally, If a var is assigned a value outside of a function it is considered global, but if var is assigned inside a function then it has a local scope. var variable can also be declared differently as well as updated.**

**What are the differences between callbacks and promises?**

**There are a lot of similarities between callbacks and promises, and promises were invented after the birth of JavaScript because developers were having issues with using excessive callbacks.**

**Callback functions operate by passing another function into the callback as an argument which is activated in the initial function to achieve an action or to establish a routine. There are two types of callbacks to be knowledgeable about: synchronous and asynchronous.**

**Promises, however, are an object that signifies the current or imminent completion or failure of an asynchronous callback.Callbacks are able to attach to a promise object in lieu of calling back to a function. What separates promises from callbacks is that they have some constants. One constant of a promise is that when added to a callback structure like then()  they won't be invoked before the completion of the current run in a JavaScript loop.**

**Sources:**

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/var**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/var)

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const)

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let)

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using\_promises**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises)

[**https://developer.mozilla.org/en-US/docs/Glossary/Callback\_function**](https://developer.mozilla.org/en-US/docs/Glossary/Callback_function)

[**https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/**](https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/)