

Javascript Concepts You Should Know



Scope

Scope determines the **accessibility of variables**, objects and functions.

- In Javascript, a variable has **3 types of scope** :
 - a. Block Scope
 - b. Function Scope
 - c. Global Scope

Hoisting

Hoisting in Javascript is a behavior in which a function or variable can be used before declaration.

- In terms of variable and constant keyword var is hoisted, and let and const does not allow hoisting.

```
// Example with var
console.log(x); // Output: undefined
var x = 5;

// Example with let
console.log(y); // Throws an error
let y = 10;

// Example with const
console.log(z); // Throws an error
const z = 15;
```

Closures

Closure means that an **inner function always has access** to the variable of **its outer function**, even after the outer function has returned.

```
function outerFunction() {  
  var outerVariable = "I'm from the outer function";  
  function innerFunction() {  
    console.log(outerVariable);  
  }  
  return innerFunction;  
}  
  
var myClosure = outerFunction();  
  
// Even though outerFunction has finished executing,  
// innerFunction still has access to outerVariable  
myClosure(); // Output: "I'm from the outer function"
```

Callbacks

A callback function can be defined as a **function passed into another function** as a parameter.

```
function greet(name, callback) {  
  console.log("Hi", name)  
  callback()  
}  
// callback function  
function callMe(){  
  console.log("I am callback funtion")  
}  
// passing function as an argument  
greet("Imtiyaz", callMe)  
  
// output  
// Hi Imtiyaz  
// I am callback funtion
```


Promises

Promises is a good way to **handle asynchronous operations**.

- It's used to **find out** if the asynchronous operation is successfully **completed or not**.
- A promise may have one of **three states**:
 - a. Pending
 - b. Fulfilled
 - c. Rejected

Async & Await

Stop and wait until something is resolved.

- We use the **async keyword** with a function to represent that the function is an **asynchronous function**.
- The async function returns a promise.

```
const showPost = async() => {  
  const res= await fetch("https://xyz.com/posts")  
  return res.posts;  
}  
  
console.log(showPost())
```