Javascript

What is hoisting : hoisting is javascript default behavior of moving declarations to the top and it will work only var

Declarations can be both var and function

Simple words javascript interpreter code execute krny say phly assume krta ahy var and function declaration jaha kahi bhi likha ho wo assume krta hay ki wo top py likha hay

Var a:

Console.log(a)

Var a=5

Array splice method: esme 2 argument hty h first means jaha say start hoga or dosra jaha tak delete hoga agar bagair const banye kiya tu estrh del hgi wrna srf wahi bachigi

* var ka scope function ya global hota hai aur ye hoisted hota hai, multiple times re-declare ho sakta hai.
* let ka scope block-level hota hai, hoisted hota hai, lekin re-declaration allowed nahi.
* const ka scope block-level hota hai, re-declaration aur re-assignment dono allowed nahi, lekin objects/arrays mein properties ya elements change kar sakte hain.

Aam tor par, modern JavaScript code mein let aur const ko prefer kiya jata hai, var ki bajaye, kyunki ye zyada predictable aur bug-free code likhne mein madad karte hain.

Lexical scope ye ensure karta hai ke functions ko apne outer scope ke variables access karne ka permission milta hai, aur yeh scope ki location ke hisaab se determine hota hai.

4o

**Summary:**

* **Closures** wo functions hain jo apne outer function ke scope ke variables ko yaad aur access kar sakte hain.
* Yeh encapsulation aur state management ko possible banate hain.
* Closures ka use private data ko manage karne, state ko track karne, aur callback functions ko implement karne mein hota hai.

Promisses

Promise aapko async tasks ka result handle karne ka ek tareeqa provide karta hai. Agar task successful hota hai, to .then() se result milta hai, aur agar fail hota hai, to .catch() se error milti ha

promise jo hay na ek object wo ascychrouns operation o handle krta hay osky k 3 states hy

pending,fullfilled, Reject

jab hum code m async lagaty hay u wo promse return hay or await async complete k wait krta hay

promise reslove hny tk wait krta hay