



Most common **JavaScript**
Interview Questions

Closures

in **JavaScript**

like and share





Closures can be confusing for newer developers, but once the concept *clicks*, they become very intuitive.



Swipe →

So what are **closures** and why should you bother?

Closure is defined as an **inner function** that has **access** to variables and parameters of an outer function even **after the outer function has returned!!**



In simple words, closures are Javascript **functions** combined with its **lexical environment**.



```
let shield = 50; } Parent scope
```

```
function runtimeTerror(health) {  
    return health + shield;  
}
```



captures from
parent scope

The **lexical environment** is generally an outer function that provides its local variables and other information to the inner function.



```
function outerFunction() {  
    let outerVar = 'I am outside!';  
    function innerFunction() {  
        console.log(outerVar);  
        // "I am outside!"  
    }  
    return innerFunction;  
}  
function execute() {  
    const myInnerFunction = outerFunction();  
    myInnerFunction();  
}  
execute();
```

The key to forming a closure is returning a function from another function.





But **why** use closures in Javascript?

For **data privacy** and encapsulation

Encapsulation in JavaScript is like making the properties of an object private

The **simplest** and most elegant way to do that is using closures.

Do you find it helpful?

let me know down in the
comments !



Slobodan Gajić

Content Creator



FOLLOW FOR MORE