

Closures in JS 🔥 | Namaste JavaScript Episode 10



Akshay Saini x Closures - JavaScript | MDN x +
127.0.0.1:5500

JS index is x JS closures.js <> index.html

Namaste 🙏 JavaScript

CLOSURES

{} Coverage: n/a

⏸ ⏮ ⏪ ⏩ ⏭ 🔊 ⏸

▼ Call Stack

Not paused

▼ Breakpoints

No breakpoints

▶ XHR/fetch Breakpoints

▶ DOM Breakpoints

▶ Global Listeners

Console What's New



☰ Closures in JS 🔥 | Namaste JavaScript Episode 10



Akshay Saini x Closures - JavaScript | MDN x +
127.0.0.1:5500

Namaste 🙏 JavaScript

Elements Console Sources Network >> | Settings | X

top Filter Custom levels | Settings

7 index.js:4


>

function bundled together with its lexical environment

it kinda remebers its lexical scope

JS index.js JS closures.js index.html

```
js > JS index.js
1 function x(){
2   var a= 7;
3   function y(){
4     console.log(a);
5   }
6   y();
7 }
8 x();|
```



Closures

A **closure** is the combination of a function bundled together (enclosed) with references to its surrounding state (the **lexical environment**). In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

Lexical scoping

Consider the following example code:

```
1 function init() {  
2   var name = 'Mozilla'; // name is a local variable  
3   function displayName() { // displayName() is the i  
4     alert(name); // use variable declared in the par  
5   }  
6   displayName();
```



JS index.js

JS closures.js

index.html

js > JS index.js

```
1 function x(){  
2   var a= 7;  
3   function y(){  
4     console.log(a);  
5   }  
6   y();  
7 }  
8 x();
```

Akshay Saini | Closures - JavaScript | MDN | 127.0.0.1:5500 | Paused in debugger

Namaste 🙏 JavaScript

Elements | Console | Sources | Network

index.js x

```
1 function x(){
2   var a= 7;
3   function y(){
4     console.log(a);
5   }
6   y();
7 }
8 x();
```

can pass or return a function to a function in javascript which allows us to do unbelievable things... and here closure become a bit complicated

when a function is returned, it is returned with its scope and lexical environment (closure)

Line 3, Column 16 | Coverage: n/a

Call Stack

| | |
|-------------|------------|
| y | index.js:4 |
| x | index.js:6 |
| (anonymous) | index.js:8 |

Breakpoints

- ☒ index.js:4 console.log(a);

XHR/fetch Breakpoints

Scope | Watch

Local

- this: Window

Closure (x)

- a: 7


Global

- PERSISTENT: 1
- TEMPORARY: 0
- addEventListener: f addEventListener()
- alert: f alert()
- atob: f atob()

Console | What's New

JS index.js | JS closures.js | index.html

```
js > JS index.js
1 function x(){
2   var a= 7;
3   function y(){
4     console.log(a);
5   }
6   return y;
7 }
8 x();
```



SUBSCRIBE

Akshay Saini | Closures - JavaScript | MDN | 127.0.0.1:5500

Namaste 🙏 JavaScript

Elements | Console | Sources | Network

top | Filter | Custom levels


```
f y(){  
  console.log(a);  
}  
100
```

index.js:10
index.js:4

a's reference is passed, thats why 100 is printed

JS index.js | JS closures.js | index.html

```
js > JS index.js  
1 function x(){  
2   var a= 7;  
3   function y(){  
4     console.log(a);  
5   }  
6   a = 100;  
7   return y;  
8 }  
9 var z = x();  
10 console.log(z);  
11 //.....  
12 z();  
13  
14
```



Akshay Saini | Closures - JavaScript | MDN | 127.0.0.1:5500

Namaste 🙏 JavaScript

Paused in debugger

index.js x

```
1 function z() {
2   var b = 900;
3   function x(){
4     var a= 7;
5     function y(){
6       console.log(a,b);
7     }
8     y();
9   }
10  x();
11 }
12 z();
13
```

forms closure over more deeper level also.. it simply depict that it cover the deepest lexical environment also

Line 6, Column 7 Coverage: n/a

Paused on breakpoint

Call Stack

| y | index.js:6 |
|-------------|-------------|
| x | index.js:8 |
| z | index.js:10 |
| (anonymous) | index.js:12 |

Breakpoints

Scope

- Local
 - this: Window
- Closure (x)
 - a: 7
- Closure (z)
- Global
 - PERSISTENT: 1
 - TEMPORARY: 0
 - addEventListener: f addEventListener()
 - alert: f alert()


Window

Console What's New

JS index.js JS closures.js index.html

js > JS index.js

```
1 function z() {
2   var b = 900;
3   function x(){
4     var a= 7;
5     function y(){
6       console.log(a,b);
7     }
8     y();
9   }
10  x();
11 }
12 z();
13
```



SUBSCRIBE



Elements

Console

Sources

Network



top



Filter

Custom levels

> Uses of Closures:

- Module Design Pattern
- Currying
- Functions like once
- memoize
- maintaining state in async world
- setTimeouts
- Iterators
- and many more...

