

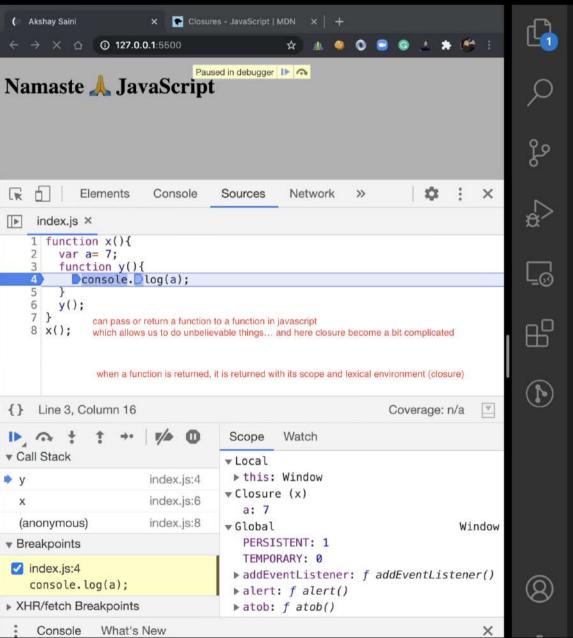
A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). In other words, a closure gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created,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

Lexical scoping

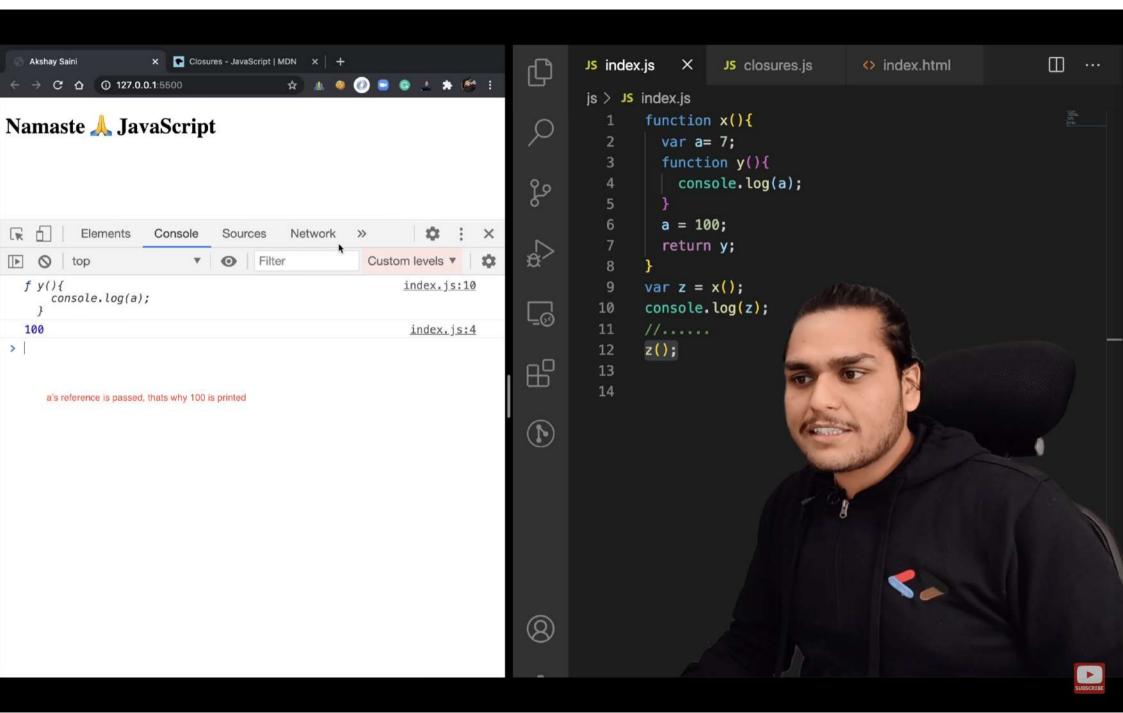
Consider the following example code:

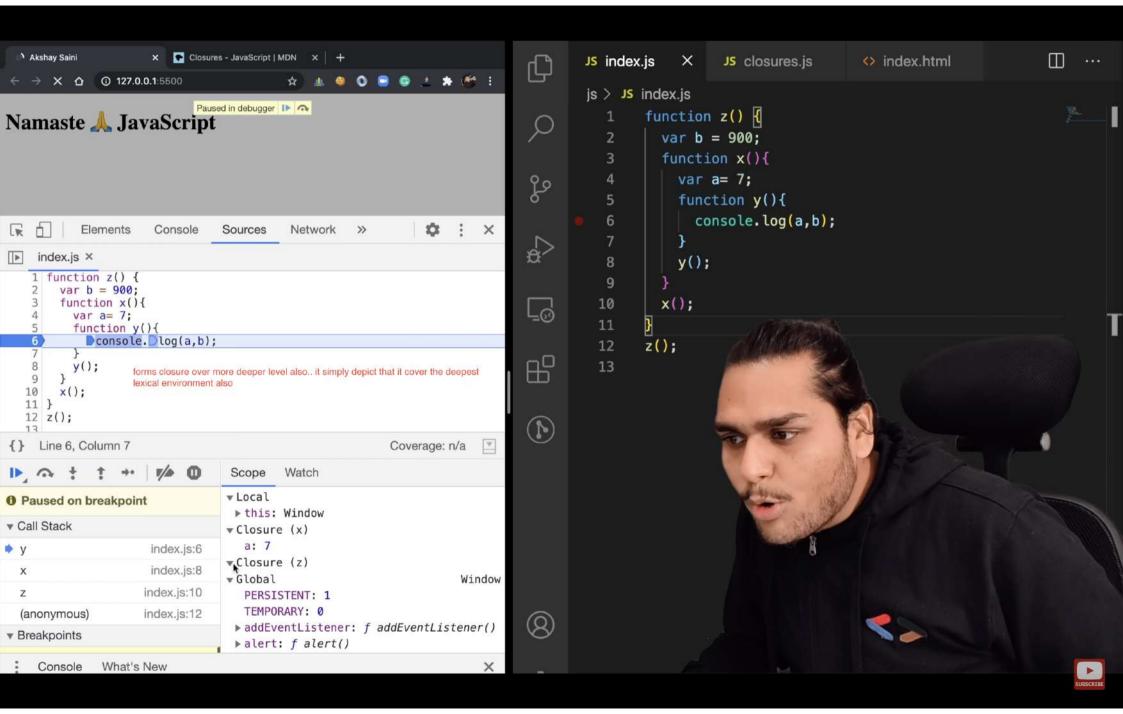
```
function init() {
  var name = 'Mozilla'; // name is a local variable
  function displayName() { // displayName() is the i
    alert(name); // use variable declared in the par
  }
  displayName();
```

```
JS index.js
                         JS closures.is
                                          index.html
       is > JS index.is
               function x(){
                 var a= 7;
          2
                 function y(){
                   console.log(a);
          6
                y();
               x();
昭
(8)
```



```
JS index.js
                  JS closures.is
                                    index.html
js > JS index.js
       function x(){
          var a= 7;
          function y()
            console.log(a);
         return y;
       x();
```









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...

