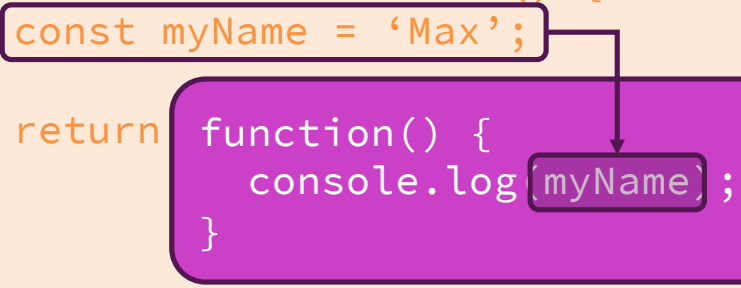


What are “Closures”?

Every function in JavaScript forms a closure. A closure is the combination of a function with its “**lexical environment**” (the variables in scope) when the function was declared.

```
function createGreeter() {  
  const myName = 'Max';  
  
  return function() {  
    console.log(myName);  
  }  
}  
  
const greet = createGreeter();  
greet(); // 'Max'
```



The anonymous function maintains a reference to its **environment**, to which “myName” belongs. Hence, access to “myName” is possible, **even after createGreeter() finished executing!**

Variables are Remembered, NOT Values

Closures maintain a **reference to the environment** – i.e. to the “**names of the available variables**”. Closures do **NOT “memorize” the values**, only the variable names. **Values are only looked up when the closure really executes** (not when it’s declared).

Summary – Closures

Every function in JavaScript is a “Closure”! Because every function in JavaScript “closes over” its environment and “memorizes” the variables that belong to that environment (i.e. that are in scope).

Very important: The variables are memorized – **NOT the values** of the variables when the closure is declared!

The values are **only retrieved when the closure (function) code is executed.**