JavaScript, HTML & Styles

1. [Difference between localStorage , sessionStorage and Cookies](#_Difference_between_localStorage,)
2. [Closures](#_Closures)

# Difference between localStorage, sessionStorage and Cookies

**Cookies**

* Has different expiration dates (both the server or client can set up expiration date)
* The Client can't access the Cookies if the HttpOnly flag is true
* Has SSL Support
* Data are transferred on each HTTP request
* 4kb limit

**Local Storage**

* Has no expiration date
* Client only
* Has no SSL support
* Data are not transferred on each HTTP request
* 5 mb limit (check with the browser)

**Session Storage**

* Data is gone when you close the browser tab
* Client only
* Has no SSL support
* Data are not transferred on each HTTP request
* 5-10 mb limit (check with the browser)

# 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.

To use a closure, define a function inside another function and expose it. To expose a function, return it or pass it to another function.

The inner function will have access to the variables in the outer function scope, even after the outer function has returned.

Example:

function outer() {

var b = 10;

var c = 100;

function inner() {

var a = 20;

console.log("a= " + a + " b= " + b);

a++;

b++;

}

return inner;

}

var X = outer(); // outer() invoked the first time

var Y = outer(); // outer() invoked the second time

//end of outer() function executions

X(); // X() invoked the first time

X(); // X() invoked the second time

X(); // X() invoked the third time

Y(); // Y() invoked the first time

When you run this code, you will see the following output in the console.log:

a=20 b=10

a=20 b=11

a=20 b=12

a=20 b=10