



**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**



-----**TECNOLOGÍAS PARA LA WEB**-----

**ACTIVIDAD:**

Strings, almacenamiento

**ALUMNO:**

Meza Vargas Brandon David – 2020630288

**GRUPO:**

2CM16

**PROFESOR:**

Rivera De La Rosa Mónica

## métodos de String

Las cadenas se pueden crear como primitivas, a partir de cadenas iterables o como objetos, usando el constructor String():

```
const string4 = new String("Un objeto String");

const string = "Cadena primitiva"
```

Las cadenas tienen ciertos métodos que nos permiten realizar ciertas operaciones con ellas, como, por ejemplo:

**-repeat():** devuelve un String con el número de copias de la cadena especificado por parámetros.

```
const string = 'Hola |'

console.log(string.repeat(5));
```

```
Hola Hola Hola Hola Hola
```

**-localeCompare():** comprueba si dos cadenas son equivalentes en la configuración regional actual. Regresa 0 si son iguales, -1 si la string 1 está ordenada antes de la string 2 y devuelve 1 si la string 1 está ordenada después de la string 2.

```
const string = 'Hola'
const string2 = 'Hola'
const string3 = 'ab'
const string4 = 'cd'
let n
n = string.localeCompare(string2)
console.log(n+"\n");
n = string3.localeCompare(string4)
console.log(n+"\n");
n = string4.localeCompare(string3)
console.log(n);
```

```
0
```

```
-1
```

```
1
```

```
>
```

**-charAt(índice):** devuelve el carácter que hay en la posición indicada como índice.

```
const string = 'Hola'
console.log(string.charAt(3));
```

a

**-indexOf(carácter, desde):** devuelve la posición de la primera vez que aparece el carácter indicado por parámetro en un string. Si no encuentra el carácter en el String devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir de que posición se desea que empiece la búsqueda.

```
const string = 'Hola'
console.log(string.indexOf('o'));
```

1

>

**-lastIndexOf(string):** devuelve la posición de la última ocurrencia del carácter pasado como parámetro.

**match(regex):** busca una coincidencia en una cadena y devuelve todas las coincidencias encontradas.

```
cadena = "Para más información, vea Capítulo 3.4.5.1";
expresion = /(capítulo \d+(\.\d)*)/i;
hallado = cadena.match(expresion);
console.log(hallado);
```

```
▼ (3) ["Capítulo 3.4.5.1", "Capítulo 3.4.5.1", ".1", index: 26, input: "Para más información, vea Capítulo 3.4.5.1", groups: undefined]
  0: "Capítulo 3.4.5.1"
  1: "Capítulo 3.4.5.1"
  2: ".1"
  groups: undefined
  index: 26
  input: "Para más información, vea Capítulo 3.4.5.1"
  length: 3
```

**-replace(cadena, sustituto):** busca una coincidencia en una cadena y si existe, la reemplaza por otra cadena pasada como parámetro.

```
const string = "Hola Jonathan"
console.log(string.replace("Hola", "Adios"));
```

Adios Jonathan

**-search(cadena):** busca una coincidencia en una cadena y devuelve la posición de la coincidencia

```
const string = "Hola Jonathan"
console.log(string.search("Jonathan"));
```

5

**-slice(inicio, final):** extrae una parte de una cadena en base a los parámetros que indiquemos como índices de inicio y final.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1, 3);
console.log(citrus);
```

▶ (2) ["Orange", "Lemon"]

**-split(separador):** corta una cadena en base a un separador que pasamos como parámetro.

```
const string = "Hola amigo como andas?"
console.log(string.split(" "));
```

▶ (4) ["Hola", "amigo", "como", "andas?"]

**-substr(inicio, longitud):** devuelve una subcadena en base a un índice y longitud pasados como parámetros.

```
var str = "Hola mundo!";
console.log(str.substr(1, 4));
```

ola

**-substring(inicio, fin):** devuelve una subcadena en base a un índice de inicio y de final pasados como parámetros.

```
var str = "Hola mundo!";
console.log(str.substring(1, 4));
```

ola

**-toLowerCase() y toUpperCase():** en el caso de la primera devuelve la cadena en minúsculas y la segunda en mayúsculas, respectivamente.

```
const string = 'HOLA'
const string2 = 'adios'
console.log(string.toLocaleLowerCase());
console.log(string2.toLocaleUpperCase());
```

hola

ADIOS

**-trim():** elimina los espacios del principio y final del string.

```
const string = "      Hola      "  
console.log(string.trim());
```

Hola

**-concat():** une dos o más Strings y los devuelve concatenados en uno nuevo.

```
const string = "Hola "  
const string2 = "amigo "  
const string3 = string.concat(string2)  
console.log(string3);
```

Hola amigo

**-includes(Cadena):** comprueba si el string contiene la cadena pasada por parámetro.

```
const string = "Hola amigo como estas"  
console.log(string.includes("como"));  
console.log(string.includes("maestra"));
```

true  
false

## -----Cookies-----

Una cookie es un fragmento de información que un navegador web almacena en el disco duro del visitante a una página web. La información se almacena a petición del servidor web, ya sea directamente desde la propia página web con JavaScript o desde el servidor web mediante las cabeceras HTTP, que pueden ser generadas desde un lenguaje de web scripting como PHP. La información almacenada en una cookie puede ser recuperada por el servidor web en posteriores visitas a la misma página web.

Las cookies resuelven un grave problema del protocolo HTTP: al ser un protocolo de comunicación "sin estado" (*stateless*), no es capaz de mantener información persistente entre diferentes peticiones. Gracias a las cookies se puede compartir información entre distintas páginas de un sitio web o incluso en la misma página web pero en diferentes instantes de tiempo.

Veamos un ejemplo para la creación de una cookie:

Para crear una cookies usamos una cadena que defina la cookie y asignarlo a document.cookie:

```
const expiresdate = new Date(2050, 1, 02, 11, 20)
const valor = "Primera cookie"
document.cookie = "testcookie="+ encodeURIComponent(valor + ";expires="+expiresdate.toUTCString())
```

Si vemos las cookies en nuestro navegador veremos la cookie creada:

|            |                  |           |             |           |
|------------|------------------|-----------|-------------|-----------|
| testcookie | Primera%20cookie | 127.0.0.1 | /string.... | 2050-0... |
| testcookie | verde            | 127.0.0.1 | /string.... | 2021-0... |
| Primera    | primera          | 127.0.0.1 | /string.... | 2021-0... |

Para modificar una cookie solo basta con cambiar el valor de una cookie anteriormente creada:

```
const expiresdate = new Date(2050, 1, 02, 11, 20)
const valor = "Primera cookie"
document.cookie = "testcookie="+ encodeURIComponent(valor) + "; expires="+expiresdate.toUTCString()
document.cookie = "testcookie="+ encodeURIComponent("hola") + "; expires="+expiresdate.toUTCString()
```

|            |         |           |             |           |
|------------|---------|-----------|-------------|-----------|
| testcookie | hola    | 127.0.0.1 | /string.... | 2050-0... |
| color      | verde   | 127.0.0.1 | /string.... | 2021-0... |
| Primera    | primera | 127.0.0.1 | /string.... | 2021-0... |

Para eliminar una cookie desde javascript debemos asignar una fecha de caducidad (expires) pasada o un max-age igual a cero

```
//eliminando cookie
document.cookie = "testcookie=; max-age= 0"
```

Y si vamos al navegador ya no estará esa cookie:

|         |         |           |             |           |
|---------|---------|-----------|-------------|-----------|
| color   | verde   | 127.0.0.1 | /string.... | 2021-0... |
| Primera | primera | 127.0.0.1 | /string.... | 2021-0... |

Para obtener el valor y leer los datos de una cookie es algo tedioso, pues solo podemos obtener un string con todas las cookies válidas para el documento y manipular el string hasta encontrar el nombre y valor de la cookie que queremos.

Se obtienen de la siguiente forma:

```
//obteniendo cookies
var lasCookies = document.cookie;
console.log(lasCookies);
```

```
Primera=primera; color=verde
```

## -----Local Storage y Session Storage-----

localStorage y sessionStorage son propiedades que acceden al objeto Storage y tienen la función de almacenar datos de manera local, la diferencia entre éstas dos es que localStorage almacena la información de forma indefinida o hasta que se decida limpiar los datos del navegador y sessionStorage almacena información mientras la pestaña donde se esté utilizando siga abierta, una vez cerrada, la información se elimina.

Ahora veamos un ejemplo de estos dos para guardar, recuperar y vaciar el storage;

Para guardar datos podemos hacer lo siguiente:

localStorage:

```
//usando setItem
localStorage.setItem("Nombre", "Brandon");
//usando .name
localStorage.Apellido = "Meza";
```

sessionStorage:

```
//usando setItem
sessionStorage.setItem("Nombre", "Brandon");
//usando .name
sessionStorage.Apellido = "Meza";
```

Como se ve, son muy similares para guardar datos, si vamos a nuestro navegador, podemos ver estos datos:

localStorage:

| Storage               |         |
|-----------------------|---------|
| Storage               |         |
| Local Storage         |         |
| http://127.0.0.1:5500 |         |
| Key                   | Value   |
| Apellido              | Meza    |
| Nombre                | Brandon |

sessionStorage:

| Almacenamiento de sesión            |         |
|-------------------------------------|---------|
| http://127.0.0.1:5500               |         |
| IndexedDB                           |         |
| Web SQL                             |         |
| Cookies                             |         |
| http://127.0.0.1:5500               |         |
| Test Tokens                         |         |
| Key                                 | Value   |
| Apellido                            | Meza    |
| IsThisFirstTime_Log_From_LiveServer | true    |
| Nombre                              | Brandon |

Para recuperar información igual tenemos dos formas de hacerlo:

localStorage:

```
// Obtenemos los datos y los almacenamos en variables
let nombre = localStorage.getItem("Nombre"),
    apellido = localStorage.getItem("Apellido");

console.log(`Soy ${nombre} ${apellido} con localStorage`);
```

Soy Brandon Meza con localStorage

sessionStorage:

```
let nombre2 = sessionStorage.getItem("Nombre"),
    apellido2 = sessionStorage.getItem("Apellido");

console.log(`Soy ${nombre2} ${apellido2} con sessionStorage`);
```

Soy Brandon Meza con sessionStorage

Para eliminar solo un elemento dentro del storage usamos:

localStorage:

```
//quitando del storage
localStorage.removeItem('Apellido')
```

Y si vemos en el navegador ya no está ese item:

|                       |        |         |
|-----------------------|--------|---------|
| Local Storage         | Filter |         |
| http://127.0.0.1:5500 | Key    | Value   |
| Local Storage         | Nombre | Brandon |
| http://127.0.0.1:5500 |        |         |

sessionStorage:

```
sessionStorage.removeItem('Apellido')
```

Y si vemos en el navegador ya no está ese item:

|                          |                                     |         |
|--------------------------|-------------------------------------|---------|
| http://127.0.0.1:5500    | Key                                 | Value   |
| Almacenamiento de sesión | IsThisFirstTime_Log_From_LiveServer | true    |
| http://127.0.0.1:5500    | Nombre                              | Brandon |

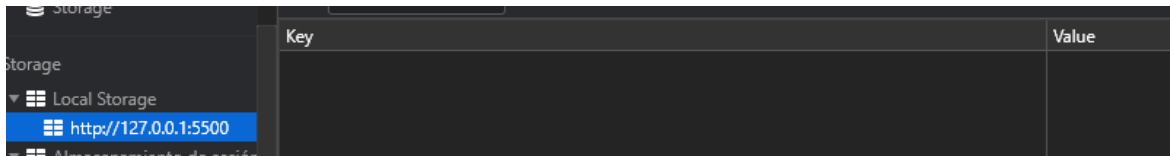


Por último, para limpiar todo el storage usamos:

localStorage:

```
//limpiando todo el storage  
localStorage.clear()
```

Si vemos en nuestro navegador ya no tendremos nada en el storage:



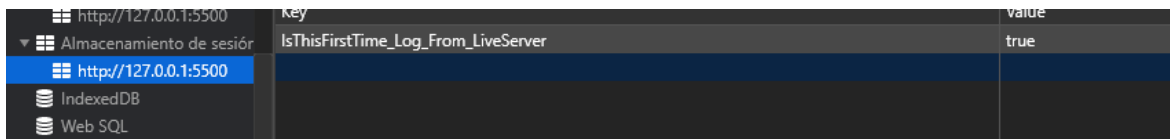
The screenshot shows the browser's storage inspector. On the left, the 'Storage' panel is expanded, showing 'Local Storage' for the URL 'http://127.0.0.1:5500'. The main area displays a table with two columns: 'Key' and 'Value'. The table is currently empty, indicating that all local storage data has been cleared.

| Key | Value |
|-----|-------|
|-----|-------|

sessionStorage:

```
sessionStorage.clear()
```

Si vemos en nuestro navegador ya no tendremos nada en el storage:



The screenshot shows the browser's storage inspector. On the left, the 'Storage' panel is expanded, showing 'Almacenamiento de sesión' (Session Storage) for the URL 'http://127.0.0.1:5500'. The main area displays a table with two columns: 'Key' and 'Value'. The table is currently empty, indicating that all session storage data has been cleared.

| Key | Value |
|-----|-------|
|-----|-------|