**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.

Pantalla de un celular con letras

Descripción generada automáticamente con confianza media



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

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



**-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 Sting devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir de que posición se desea que empiece la búsqueda.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



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

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

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

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





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





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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



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





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





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





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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



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

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



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

Texto

Descripción generada automáticamente



-------------------------------------------------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:

Imagen de la pantalla de un celular de un mensaje en letras blancas

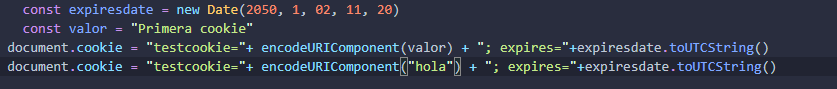
Descripción generada automáticamente con confianza baja

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

Patrón de fondo

Descripción generada automáticamente

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



Patrón de fondo

Descripción generada automáticamente

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

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

Y si vamos al navegador ya no estará esa cookie:

Patrón de fondo

Descripción generada automáticamente

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:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



----------------------------------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:

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

sessionStorage:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

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

localStorage:

Patrón de fondo

Descripción generada automáticamente

sessionStorage:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para recuperar información igual tenemos dos formas de hacerlo:

localStorage:

Texto

Descripción generada automáticamente



sessionStorage:

Texto

Descripción generada automáticamente



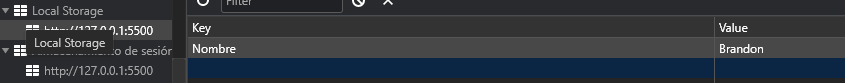
Para eliminar solo un elemento dentro del storage usamos:

localStorage:

Texto

Descripción generada automáticamente

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



sessionStorage:

Texto

Descripción generada automáticamente

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



Por último, para limpiar todo el storage usamos:

localStorage:

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

sesionStorage:

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

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

