

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 07. JAVASCRIPT

STORAGE Y JSON



OBJETIVOS DE LA CLASE

- Comprender la utilidad de localStorage y sessionStorage
- Comprender el alcance del formato JSON, y entender en qué situaciones es necesaria su utilización.

GLOSARIO:

Clase 6

Operar: en programación, cuando hablamos de operar sobre las variables, nos referimos a utilizarlas en funciones, métodos, o a lo largo del código. Consiste en desarrollar los algoritmos a partir, y en función del valor de estas variables.

Propiedad length: nos permite saber el largo de una cadena String, es decir, cuántos caracteres tiene.

Método replace (): permite reemplazar un carácter o grupo de caracteres por otros.

Método trim (): permite quitar los espacios ubicados al principio y al final de la cadena.

Array: es una variable que almacena una lista de elementos. Puede ser una lista de números, una lista de números y palabras o hasta una lista de listas.

Método slice: devuelve una copia de una parte del array dentro de un nuevo array, empezando por inicio hasta fin (fin no incluido). El array original no se modificará.

Método toString: convierte un Array a un String, compuesto por cada uno de los elementos del Array separados por comas.

Método push: se utiliza para sumar un elemento a un Array ya existente.

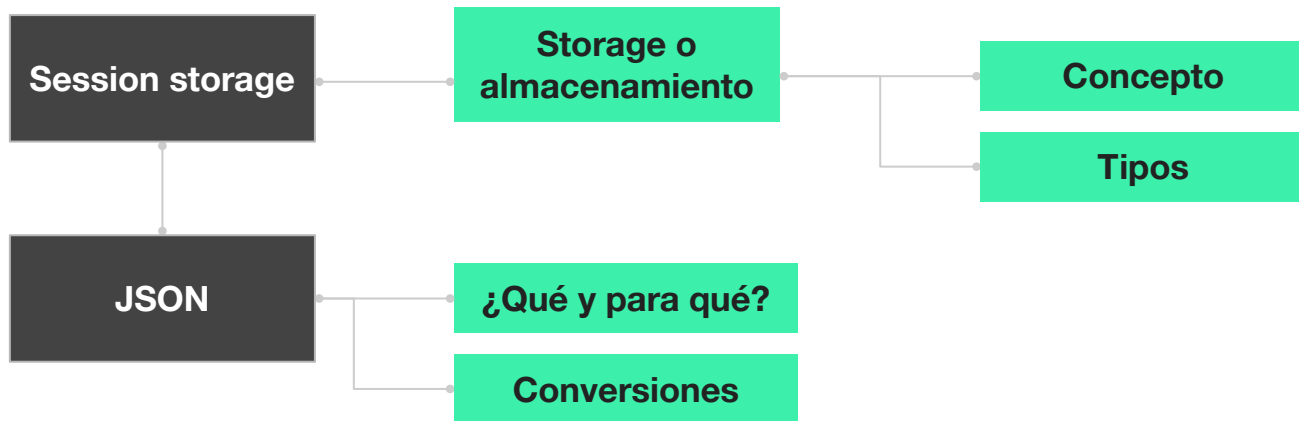
Método join: permite juntar todos los elementos de un Array en una cadena String.

Método concat: combinar dos arrays en un único array resultante.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 7

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 6



Arrays



EJEMPLOS EN VIVO



CREAR UN ALGORITMO
CON ARRAYS



INCORPORAR ARRAYS

Clase 7



Storage y JSON



EJEMPLOS EN VIVO



EJERCITAR JSON Y
STORAGE

Clase 8



DOM



EJEMPLOS EN VIVO



INTERACTUAR CON HTML

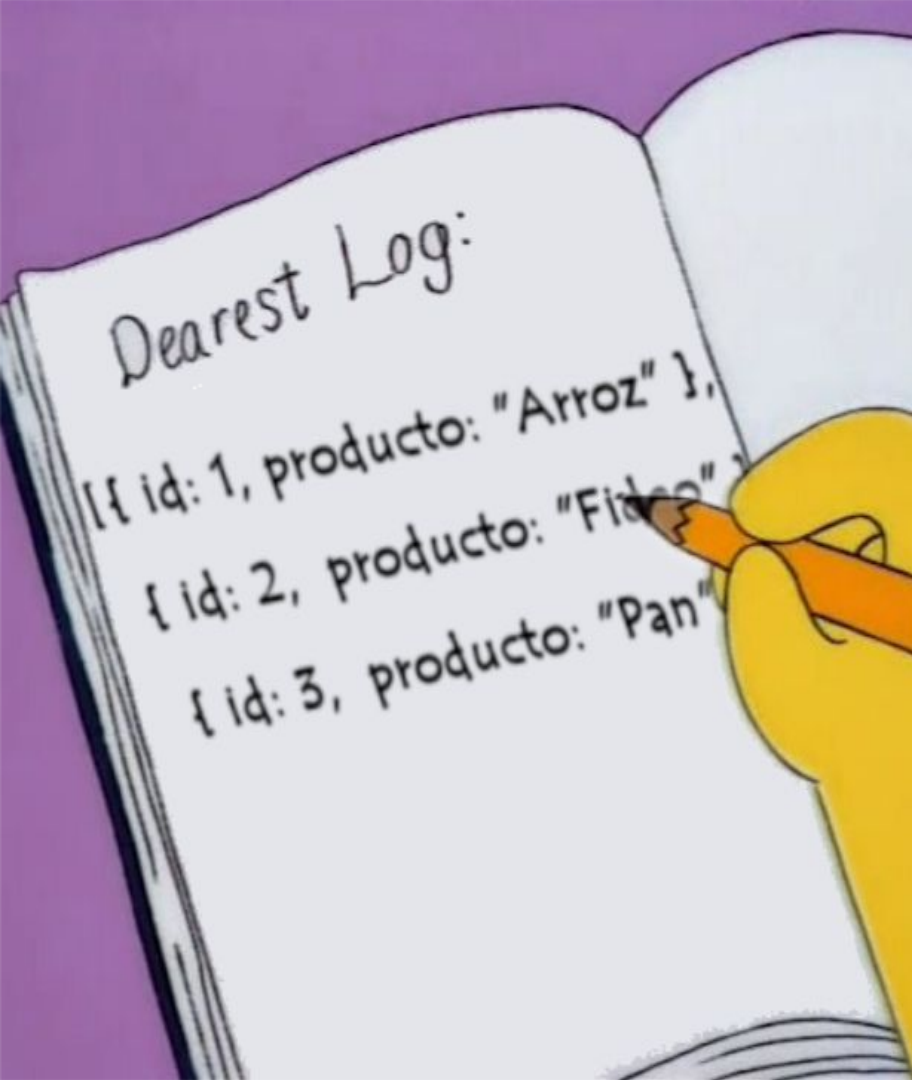


HERRAMIENTAS DE LA CLASE

Les compartimos algunos recursos para acompañar la clase

- Guión de clase N° 7 [aquí](#).
- Quizz de clase N° 7 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

STORAGE



STORAGE O ALMACENAMIENTO

El objeto Storage (API de almacenamiento web) nos permite **almacenar datos de manera local** en el navegador sin necesidad de realizar ninguna conexión con el servidor.

Lo que significa que cada cliente puede preservar información de la aplicación.

Estudiaremos dos tipos de almacenamiento: `localStorage` y `sessionStorage`. Uno es indefinido en el navegador, y otro es temporal, hasta cerrar la pestaña.

LOCALSTORAGE- SETITEM

Los datos almacenados en `localStorage` (variable global preexistente) se almacena en el navegador de forma indefinida (o hasta que se borren los datos de navegación del browser): La información persiste reinicios de navegador y hasta del sistema operativo .

Para almacenar información se utiliza `setItem`:

```
// Método -> localStorage.setItem(clave, valor)
// clave = nombre para identificar el elemento
// valor = valor/contenido del elemento
localStorage.setItem('bienvenida', '¡Hola Code!');
localStorage.setItem('esValido', true);
localStorage.setItem('unNumero', 20);
```

LOCALSTORAGE: GETITEM

Podemos acceder a la información almacenada en localStorage utilizando getItem. Las claves y valores de Storage se guardan en formato de cadena de caracteres (DOMString).

```
let mensaje = localStorage.getItem('bienvenida');  
let bandera = localStorage.getItem('esValido');  
let numero = localStorage.getItem('unNumero');  
  
console.log(typeof mensaje); //string;  
console.log(typeof bandera); //string;  
console.log(typeof numero); //string;
```

SESSIONSTORAGE- SETITEM

La información almacenada en `sessionStorage` (variable global preexistente) se almacena en el navegador hasta que el usuario cierra la ventana. Solo existe dentro de la pestaña actual del navegador y otra pestaña con la misma página tendrá otro `sessionStorage` distinto, pero se comparte entre iframes en la pestaña (asumiendo que tengan el mismo origen). Para almacenar información se utiliza `setItem`:

```
// Método -> sessionStorage.setItem(clave, valor)
// clave = nombre del elemento
// valor = Contenido del elemento
sessionStorage.setItem('seleccionados', [1,2,3]);
sessionStorage.setItem('esValido', false);
sessionStorage.setItem('email', 'info@email.com');
```

SESSIONSTORAGE: GETITEM

Podemos acceder a la información almacenada en `sessionStorage` utilizando `getItem`. Las claves y valores de Storage se guardan siempre en formato de cadena de caracteres.

```
let lista    = sessionStorage.getItem('seleccionados').split(",");
let bandera = (sessionStorage.getItem('esValido') == 'true');
let email    = sessionStorage.getItem('email');

console.log(typeof lista);    //object ["1","2","3"];
console.log(typeof bandera); //boolean;
console.log(typeof email);    //string;
```

ACCESO TIPO OBJETO

Dado que `localStorage` y `sessionStorage` son objetos globales es posible crear y acceder a las claves como si fueran propiedades. Pero esto no es recomendable, porque hay eventos asociados a la modificación del storage cuando se emplea `getItem` o `setItem`

```
//Guarda una clave
localStorage.numeroPrueba = 5;

//Leer una clave
alert( localStorage.numeroPrueba ); // 5

let clave = 'toString'; //toString método reservado
localStorage[clave] = "6"; //No se guarda este dato
```


RECORRIENDO EL STORAGE

Es posible obtener todos los valores almacenados en `localStorage` o `sessionStorage` con un bucle. Pero no podemos usar `for...of` porque no son objetos iterables, ni `for...in` porque obtenemos otras propiedades del objeto que no son valores almacenados. El bucle a emplear es `for` con el método `key`

```
//Ciclo para recorrer las claves almacenadas en el objeto localStorage
for (let i = 0; i < localStorage.length; i++) {
    let clave = localStorage.key(i);
    console.log("Clave: " + clave);
    console.log("Valor: " + localStorage.getItem(clave));
}
```

ELIMINAR DATOS DEL STORAGE

Podemos eliminar la información almacenada en `sessionStorage` o `localStorage` usando el método `removeItem` o `clear`:

```
localStorage.setItem('bienvenida', '¡Hola Code!');  
sessionStorage.setItem('esValido', true);  
  
localStorage.removeItem('bienvenida');  
sessionStorage.removeItem('esValido');  
  
localStorage.clear()    //elimina toda la información  
sessionStorage.clear() //elimina toda la información
```

EJEMPLO APLICADO: ALMACENAR TABLA DE MULTIPLICAR

```
const multiplicar = (a, b) => a * b;
const guardarLocal = (clave, valor) => { localStorage.setItem(clave, valor) };

// Solicitamos un valor al usuario
let ingresarNumero = parseInt(prompt("Ingresar Numero"));
// En cada repetición calculamos el número ingresado por el número de repetición (i)
for (let i = 1; i <= 10; i++) {
    guardarLocal(i,multiplicar( parseInt(ingresarNumero),i));
}
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

JSON

ALMACENAR OBJETOS EN STORAGE

Si queremos almacenar la información de un objeto en un storage, hay que tener en cuenta que tanto la clave como el valor se almacenan en strings. Ante cualquier otro tipo a guardar, como un número o un objeto, se convierte a cadena de texto automáticamente. Entonces al buscar almacenar un objeto, sin una transformación previa, guardamos [object Object], la conversión por defecto de objeto a string. Para guardar la información correctamente hay que transformar el objeto a **JSON**.

```
const producto1 = { id: 2, producto: "Arroz" };  
localStorage.setItem("producto1", producto1); // Se guarda [object Object]
```

¿QUÉ ES JSON?

JavaScript Object Notation (JSON) es un **formato basado en texto plano**, para representar datos estructurados con la sintaxis de objetos de JavaScript. Es comúnmente utilizado para enviar y almacenar datos en aplicaciones web.

¿QUÉ ES JSON?

Aunque es muy parecido (casi similar) a la sintaxis de JavaScript, puede ser utilizado independientemente de JavaScript, y muchos entornos de programación poseen la capacidad de leer (convertir; parsear) y generar JSON.

JSON es un string con un formato específico

CONVERSIONES DE/HACIA JSON

Cuando sea necesario enviar un objeto Javascript al servidor o almacenarlo en storage, será necesario convertirlo a un JSON (una cadena) antes de ser enviado.

- **stringify()**: acepta un objeto como parámetro, y devuelve la forma de texto JSON equivalente.
- **parse()**: recibe un texto JSON como parámetro, y devuelve el objeto JavaScript correspondiente.

STRINGIFY

Con `JSON.stringify` podemos transformar un objeto JavaScript a un string en formato JSON.

```
const producto1 = { id: 2, producto: "Arroz" };  
const enJSON    = JSON.stringify(producto1);  
  
console.log(enJSON); // {"id":2,"producto":"Arroz"}  
console.log(typeof producto1); // object  
console.log(typeof enJSON);    // string  
  
localStorage.setItem("producto1", enJSON);  
// Se guarda {"id":2,"producto":"Arroz"}
```

PARSE

Con `JSON.parse` podemos transformar string en formato JSON a objeto JavaScript.

```
const enJSON      = '{"id":2,"producto":"Arroz"}';
const producto1 = JSON.parse(enJSON);

console.log(typeof enJSON);      // string
console.log(typeof producto1);   // object
console.log(producto1.producto); // Arroz

const producto2 = JSON.parse(localStorage.getItem("producto1"));
console.log(producto2.id);       // 2
```

EJEMPLO APLICADO: ALMACENAR ARRAY DE OBJETOS

```
const productos = [{ id: 1, producto: "Arroz", precio: 125 },  
                    { id: 2, producto: "Fideo", precio: 70 },  
                    { id: 3, producto: "Pan" , precio: 50},  
                    { id: 4, producto: "Flan" , precio: 100}];
```

```
const guardarLocal = (clave, valor) => { localStorage.setItem(clave, valor) };
```

```
//Almacenar producto por producto
```

```
for (const producto of productos) {  
    guardarLocal(producto.id, JSON.stringify(producto));  
}
```

```
// o almacenar array completo
```

```
guardarLocal("listaProductos", JSON.stringify(productos));
```

EJEMPLO APLICADO: OBTENER ARRAY ALMACENADO

```
class Producto {
  constructor(obj) {
    this.nombre = obj.producto.toUpperCase();
    this.precio = parseFloat(obj.precio);
  }
  sumaIva() {
    this.precio = this.precio * 1.21;
  }
}

//Obtenemos el listado de productos almacenado
const almacenados = JSON.parse(localStorage.getItem("listaProductos"));
const productos = [];
//Iteramos almacenados con for...of para transformar todos sus objetos a tipo producto.
for (const objeto of almacenados)
  productos.push(new Producto(objeto));
//Ahora tenemos objetos productos y podemos usar sus métodos
for (const producto of productos)
  producto.sumaIva();
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

JSON

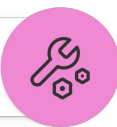
Otros puntos a tener en cuenta

- Las datos en formato JSON se pueden almacenar en archivos externos .json. Exemplo: datos.json
- JSON es sólo un formato de datos — contiene sólo propiedades, no métodos.
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione. Se debe ser cuidadoso para validar cualquier dato que se quiera utilizar.
<https://jsonformatter.curiousconcept.com/>
- A diferencia del código JavaScript en que las propiedades del objeto pueden no estar entre comillas, en JSON sólo las cadenas entre comillas pueden ser utilizadas como propiedades.



EJERCITAR JSON Y STORAGE

Realiza un algoritmo que almacene información en Storage y guarde un array de objetos en formato JSON.



¡A PRACTICAR!

Deberá cumplir los siguientes requisitos:

- Almacenar en Storage información ingresada por el usuario.

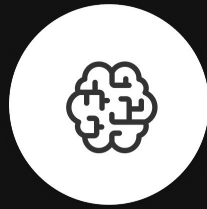
Puede ser un texto, números, o combinación. Luego mostrarla mediante alert o console.

- Declarar un array de objetos (literales, con función constructora o con clases) y almacenar el array en formato JSON en el storage.

Cuentas con 40 minutos para hacer la actividad.

¿PREGUNTAS?





¡PARA PENSAR!

¿Te gustaría comprobar tus conocimientos de la clase?

Te compartimos a través del chat de zoom
el enlace a un breve quiz de tarea.

Para el profesor:

- *Acceder a la carpeta "Quizzes" de la camada*
 - *Ingresar al formulario de la clase*
 - *Pulsar el botón "Invitar"*
 - *Copiar el enlace*
- *Compartir el enlace a los alumnos a través del chat*



RECURSOS:

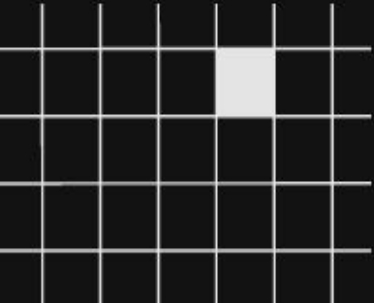
- LocalStorage, sessionStorage |
[***Javascript.info***](#)
- JSON |
[***GitBooks. El formato JSON***](#)
[***JSON Formatter***](#)
[***Generador JSON***](#)
- Documentación |
[***Documentación localStorage***](#)
[***Documentación sessionStorage***](#)
[***Documentación JSON***](#)

Disponible en [nuestro repositorio](#).



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Objeto Storage: localStorage y sessionStorage.
 - JSON: concepto y uso en JS.
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE