



Local Storage

Almacenar información en el navegador



¿Qué es el Local Storage?

El **Local Storage** es una funcionalidad de los navegadores que permite a los desarrolladores guardar información en el navegador del usuario.

Funciona guardando información en un formato de pares **clave-valor**.




Las **claves** (que son equivalentes a nombres de atributos de un objeto) y los **valores** deberán ser siempre guardados como cadenas de texto (string).

La información guardada se conserva aun cuando el usuario actualice la página o cierre la pestaña o el navegador.

El **Local Storage** no tiene fecha de expiración, por lo que los datos se mantienen hasta que se eliminan explícitamente.



Compatibilidad con navegadores

													
	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS	Deno
<code>localStorage</code>	✓ 4	✓ 12	✓ 3.5	✓ 10.5	✓ 4	✓ 18	✓ 4	✓ 11	✓ 3.2	✓ 1	✓ 4.4	✓ 3.2	✓ 1.16 *

Tip: you can click/tap on a cell for more information.

✓ Full support * See implementation notes.



Guardar datos en Local Storage

Para almacenar información en el local storage, se usa en el método `localStorage.setItem("clave", "valor")`.

Este método lleva dos argumentos, un clave y un valor (**ambos string**).

Ejemplo:

```
localStorage.setItem("tema", "dark")
```



Si la clave no existe en el **Local Storage**, el método `setItem()` va a crear una nueva clave y asignarle el valor dado.

Pero si en el **Local Storage** ya existe una clave con el mismo nombre, el valor va a ser reemplazado con el nuevo valor.

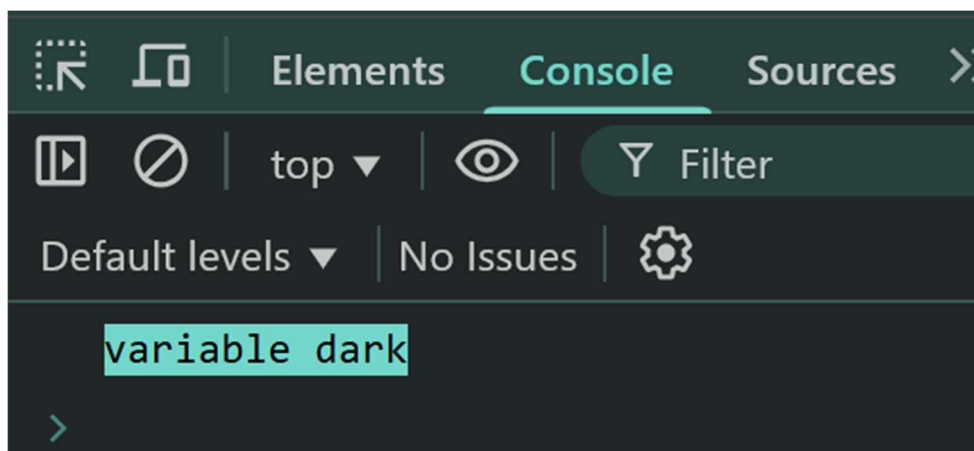


Leer información almacenada en el Local Storage

Para recuperar y usar la información del **Local Storage**, se utiliza el método `localStorage.getItem("clave")`.

Ejemplo:

```
let variable = localStorage.getItem("tema")
console.log("variable", variable) // variable dark
```



Si la clave existe en el **Local Storage**, el método devuelve el valor de esa clave.

En caso contrario, devuelve `null`



Guardar *arrays* y objetos en el Local Storage

En el **Local Storage** solo se pueden almacenar cadenas de texto.

Esto implica que si necesitamos almacenar valores como objetos o arrays en el mismo, hay que transformar ese tipo de variables en cadenas de texto (string).

Para ello tenemos el método `JSON.stringify()`.

Ejemplo:

```
const tamagochi = {  
  nombre: 'Pipo',  
  salud: 9,  
  felicidad: 10,  
  limpieza: 8,  
  energia: 9  
}
```

```
localStorage.setItem('pipo', JSON.stringify(tamagochi));
```

El método `JSON.stringify()` convierte al objeto `tamagochi` en una cadena de texto antes de enviarlo al local storage.



Recuperar *arrays* y objetos del Local Storage

Para recuperar la información de un array u objeto del **Local Storage**, es necesario devolverla a su forma original.

Esto se hace con el método `JSON.parse()`

Ejemplo:

```
let storedTamagochi = localStorage.getItem('pipo')

if (storedTamagochi) {
  tamagochi = JSON.parse(storedTamagochi)
} else {
  console.log('pipo no está en el Local Storage')
}
```

Es importante verificar si hay información para la clave **"pipo"** en local storage antes de usar el método `JSON.parse()`.

De lo contrario, el método `JSON.parse()` se le aplicará a un valor nulo lo cual produce un error.



Borrar información o vaciar el Local Storage desde JavaScript

► Para borrar información del **Local Storage** usaremos el método `localStorage.removeItem("clave")`.

El método `removeItem()` lleva la clave como argumento y borra el correspondiente par clave-valor del local storage.

Ejemplo:

```
localStorage.removeItem("tema")
```

► Para vaciar por completo el **Local Storage** usaremos el método `localStorage.clear()`.

Estos dos métodos eliminan las entradas al **Local Storage** desde un dominio y al navegador que está siendo utilizado.

¿Un usuario puede eliminar datos del Local Storage?

Un usuario puede eliminar el contenido del **Local Storage**.

Para hacerlo manualmente, se borra la caché del navegador o se selecciona la opción de **Borrar datos del sitio** en la configuración del navegador.

En algunos navegadores, como *Google Chrome*, esto se puede encontrar en Más herramientas > Borrar datos de navegación

Por lo tanto, a pesar de que permite la persistencia de los datos, el Local Storage no debe usarse como una herramienta para mantener los datos a través del tiempo, **ni para guardar datos sensibles**.



Cómo ver lo que guarda el Local Storage desde la ventana **Inspeccionar** (F12)

The screenshot shows the Chrome DevTools 'Application' tab. The left sidebar lists the 'Storage' section, with 'Local storage' selected. The main panel shows the 'file://' origin with a table of local storage items. A red box highlights the 'Local storage' item in the sidebar and the table containing the 'tema' key with the value 'dark'.

Key	Value
tema	dark

No value selected
Select a value to preview



Pasar información de un tema con Local Storage

Ejemplo con "Los pueblos más bonitos de Catalunya"



```
<head>
  <link rel="stylesheet" href="./css/normalize.css">
  <link id="theme" rel="stylesheet" href="./css/dark.css">
  <link rel="stylesheet" href="./css/estilos.css">
</head>
<body>

  <label for="tema-sel">Tema</label>
  <select id="tema-sel" onchange="cambiarTema(this.value)">
    <option value="dark">dark</option>
    <option value="light">light</option>
    ...
  </select>
  <script>
    let tema = localStorage.getItem("tema")
    if (tema) {
      document.getElementById('theme').href="./css/" + tema + ".css"
    }
    function cambiarTema(tema){
      document.getElementById('theme').href="./css/" + tema + ".css"
      localStorage.setItem("tema", tema)
    }
  </script>
```



alcanar.html

```
<head>
  <link rel="stylesheet" href="./css/normalize.css">
  <link id="theme" rel="stylesheet" href="./css/dark.css">
  <link rel="stylesheet" href="./css/estilos.css">
</head>

<script>
  let tema = localStorage.getItem("tema")
  if (tema) { // si existe la clave
    document.getElementById('theme').href="./css/" + tema + ".css"
  }
</script>
```

tema1: **light.css**

```
:root {  
  --background-color: white;  
  --text-color: blue;  
  --align-text: center;  
}
```

tema2: **dark.css**

```
:root {  
  --background-color: black;  
  --text-color: white;  
  --align-text: left;  
}
```

ccs3: **estilos.css**

```
body {  
  background-color: var(--background-color);  
  color: var(--text-color);  
}  
  
h1 {  
  text-align: var(--align-text, center);  
}
```

Lee las variables del tema