UNIDAD 9 (3) - Formularios - Almacenar datos en el equipo cliente

Desarrollo Web en Entorno Cliente

2º DAW

Contenidos

Almacenar datos en el equipo cliente

Cookies

Con `document.cookie` se obtienen y establecen las cookies asociadas al documento.
Las cookies son datos almacenados en nuestro ordenador, en pequeños archivos de texto.
Van ligadas a los navegadores y a los documentos: las que vemos en un navegador, no se pueden acceder desde otro.
Pueden tener fecha de expiración. Si no la tienen, expiran al acabar la sesión.

Leer todas las cookies accesibles

```
let todasCookies = document.cookie;
```

En el código anterior, `todasCookies` tendrá una cadena formada por una lista de todas las cookies (con formato clave=valor) separadas por punto y coma.

Ejemplo: Esta es la cadena obtenida al ejecutar el código anterior con https://devdocs.io:

```
"_gauges_unique_month=1; _gauges_unique_year=1;
docs=css/dom/dom_events/html/http/javascript; schema=2;
_ga=GA1.2.135871839.1553426731; _gid=GA1.2.1768504745.1554392662;
count=5; news=1537660800000; version=1548019000;
_gauges_unique_hour=1; _gauges_unique_day=1; _gauges_unique=1"
```

Ejemplos:

```
alert(document.cookie);
document.cookie = "nombre = Ada";
```

Crear una nueva cookie

```
document.cookie = nuevaCookie;
```

En el código anterior, `nuevaCookie` es una cadena con la forma `clave = valor`.
Solo se puede establecer/actualizar una cookie de cada vez utilizando este método.

Después del valor de la cookie se pueden especificar algunos atributos de la cookie. Se escriben con un punto y coma y después el nombre del atributo y su valor:

- `path = path;` (ej. path='/'). Si no se especifica, se toma como valor por defecto el path del documento actual.
- `max-age = max-age-en-segundos;`
- `expires = fecha-en-formato-GMT;`
  Si no es especifican `max-age` o `expires`, la cookie expira al terminar la sesión.

Ejemplo:

```
document.cookie = "nombre = pepe; max-age = 3600";
```

Modificar una cookie
Modificar una cookie es sobrescribirla.

Ejemplo:

```
document.cookie = "nombre = Laura";
```

Borrar una cookie

Para borrar una cookie se puede dar una fecha de expiración anterior a la actual.

Ejemplo:

```
document.cookie = "nombre =; expires=Thu, 01 Jan 1970 00:00:00 UTC";
```

Obtener el valor de una cookie

Para obtener el valor de una cookie individual, hay que:

- Obtener una cadena con todas las cookies y sus valores (document.cookie).
- Localizar, en esa cadena, la cookie que nos interesa y su valor.

Ejemplo[1]: Asignar a `valorCookie` el valor de la cadena con nombre `miCookie`

```
let valorCookie = document.cookie
  .split('; ')
  .find(cadena => cadena.startsWith(miCookie='))
  .split('=')[1];
```

[1] https://devdocs.io/dom/document/cookie

Web Storage

JavaScript soporta persistencia de datos en el navegador a través de Web Storage. Esta técnica, nos permite almacenar más información y de forma más intuitiva y segura que con las cookies. Otra ventaja de Web Storage es que almacena los datos por origen, es decir, todas las páginas con el mismo origen (protocolo, dominio y puerto), podrán acceder a los mismos datos. Por estos motivos, hoy en día **se recomienda utilizar esta técnica en vez de cookies**.

Disponemos de dos mecanismos distintos para trabajar con Web Storage, ambos disponibles con las siguientes propiedades del objeto `Window`:

- **`localStorage`**: permite crear contenedores de datos permanentes que solo se eliminan si se borran desde JavaScript.
- **`sessionStorage`**: permite crear contenedores de datos asociados a la sesión.
  - Comienzo de sesión: apertura de navegador o pestaña.
  - Final de sesión: cierre de navegador o pestaña.

Ambas propiedades deben almacenar siempre strings, no pueden almacenar otro tipo de datos.

Los métodos que se pueden utilizar para gestionar los datos almacenados son:

Establecer un item: .setItem()

```
localStorage.setItem("nombre", "valor");
sessionStorage.setItem("nombre", "valor");
```

También se puede hacer mediante propiedades dinámicas, pero hoy en día no se recomienda:

```
localStorage.nombre = "valor";
sessionStorage.nombre = "valor";
```

Obtener el valor de un ítem: .getItem()

```
localStorage.getItem("nombre");
sessionStorage.getItem("nombre");
```

También se puede hacer mediante propiedades dinámicas, pero hoy en día no se recomienda:

```
localStorage.nombre;
sessionStorage.nombre;
```

Eliminar un ítem: .removeItem()

```
localStorage.removeItem("nombre");
sessionStorage.removeItem("nombre");
```

Eliminar todos los ítems: .clear()

```
localStorage.clear();
sessionStorage.clear();
```

Comprobar si el navegador soporta Storage

```
if (typeof(Storage) != "undefined") {
      //Soporta Web Storage
}
```

Same Origin Policy

Hemos de tener en cuenta que los contenedores de `localStorage` y `sessionStorage` siguen la política de seguridad **same-origin-policy**, es decir, un script solo puede acceder a contenedores creados por otros scripts que vinieron del mismo origen, es decir, del mismo servidor. El origen de un script son el protocolo, dominio y puerto del servidor.

La siguiente tabla[2] muestra ejemplos de comparaciones de origenes para la URL `http://store.company.com/dir/page.html`:

URL	Resultado	Razón
http://store.company.com/dir2/other.html	Mismo origen	Solo la ruta difiere
http://store.company.com/dir/inner/another.html	Mismo origen	Solo la ruta difiere
https://store.company.com/secure.html	Fallo	Diferente protocolo
http://store.company.com:81/dir/etc.html	Fallo	Diferente puerto
http://news.company.com/dir/other.html	Fallo	Diferente host

Si se viola esta política de seguridad, el navegador lanzará la excepción ***security error*** o simplemente no funcionará correctamente. Esto suele ocurrir siempre que trabajemos con archivos locales. En ocasiones lo podemos solucionar accediendo por medio de localhost en vez de a través de la ruta a nuestro archivo, pero la solución real es alojar nuestros archivos en servidores, por ejemplo: https://neocities.org/ nos ofrece servicio gratuito.

También hemos de tener en cuenta que el usuario puede tener su navegador configurado para denegar permisos a datos persistentes para el origen especificado, en ese caso no podemos hacer nada, salvo pedirle que los permita.

[2]Tomada de: https://developer.mozilla.org/es/docs/Web/Security/Same-origin_policy