I.E.S. PALOMERAS VALLECAS

ALMACENAMIENTO DE DATOS EN EL LADO DEL CLIENTE

UT8.- ALMACENAMIENTO DE DATOS EN EL LADO DEL CLIENTE

Susana López Luengo

ALMACENAMIENTO DE DATOS

- Las aplicaciones web pueden almacenar datos localmente en el navegador del usuario con el almacenamiento local
- Antes de HTML5, los datos debían guardarse en **cookies**, incluidas en cada petición del servidor.
- El almacenamiento local es más seguro, permite guardar más información, sin afectar al rendimiento del servidor ni incumplir la ley de protección de datos
- Existen dos objetos para obtener acceso local
  - **window.localStorage**: La información no expira nunca
  - **window.sessionStorage**: La información se guarda solo por una sesión

COOKIES

- Las cookies son datos, almacenados en pequeños ficheros de textos o bases de datos embebidas en el navegador web
- HTTP es un protocolo sin estado, cuando un servidor web envía un página web al navegador, la conexión finaliza, y el servidor olvida cualquier cosa sobre el usuario
- Las cookies fueron inventadas para resolver el problema de recordar información del usuario

COOKIES

- Las cookies almacenan en un par clave-valor la información

```
usuario = pepe
```

- Cuando el navegador hace una petición de una página web a un servidor, las cookies que pertenezcan a la página se añadirán a la petición.

IMPORTANTE!!

*Cuando vayas a utilizar las cookies en **Chrome**, asegúrate que trabajas con servidor (Ej. LiveServer), con file no funciona*

COOKIES. UTILIZACIÓN

- Según el Real Decreto-ley 13/2012, de 30 de marzo se publica la ley Artículo 22.2 de la Ley 34/2002 L*os prestadores de servicios podrán utilizar dispositivos de almacenamiento y recuperación de datos en equipos a condición de que los mismos hayan dado su consentimiento*
- En caso de no cumplirse podría sancionarse con una infracción leve de hasta 30.000€ o una infracción grave de hasta 150.000 €
- Ejemplo de código

COOKIES. UTILIZACIÓN

- JavaScript puede crear, leer y borrar cookies con la propiedad document.cookie.
- Creación de una cookie, se guardará en forma de pares nombre=valor:
- Se puede poner información sobre: nombre, fecha de expiración (en formato UTC); edad máxima en segundos, ruta, dominio,

```
document.cookie = "alumno=juan;”
                                (por lo menos hay que indicar nombre)
document.cookie = “usuario=pepe”; expires=Thu, 16 Jan 2020
12:00:00 UTC; path=”/”;
```

- Si creamos la cookie con más propiedades, tenemos que indicar todas las propiedades si la queremos modificar porque si no la consideraría otra cookie

COOKIES. UTILIZACIÓN

Leer una cookie

- No podemos leer una cookie concreta, para leer todas las cookies y de la cadena extraer la cookie concreta:

```
var miscookies= document.cookie;
```

Ejemplo: crear y consultar cookie w3schools

https://developer.mozilla.org/es/docs/Web/API/Document/cookie

Modificar una cookie

```
document.cookie = "alumno=alberto;”;
```

Borrar una cookie: Ponemos una fecha de expiración anterior al momento en el que estamos (en formato UTC)

```
document.cookie = "alumno=; expires=Thu, 01 Jan 1970 00:00:01 GMT";
```

Enlace a w3schools

Ejemplo de cookie para guardar un dato por tiempo limitado

```
function guardarCookie(nombre,valor,diasExpirar) {
   var fecha = new Date();
   fecha.setTime(fecha.getTime() +
                    (diasExpirar *24*60*60*1000));
   var exp = "expires="+ fecha.toUTCString();
   document.cookie =`${nombre}=${valor}; ${exp};`

}
```

Ejemplo de cookie para leer la cookie anterior si existe

//Devuelve el valor de una cookie dada

```
function obtenerCookie(nombreCookie) {
    var nombre = nombreCookie + "=";
    var cookieDescodificada = decodeURIComponent(document.cookie);
    var arrayCookie = cookieDescodificada.split(';');
    for(var i = 0; i < arrayCookie.length; i++) {
        var cookie = arrayCookie [i].trim();
        if (cookie.indexOf(nombre) == 0) {
            return cookie.substring(nombre.length, cookie.length);
        }
    }
    return "";
}
```

CARACTERÍSTICAS DE LAS COOKIES

Una cookie es sólo información en texto plano administrable por el mismo usuario y en ningún caso es código fuente interpretable.

Parte de la sencillez de las cookies genera inconvenientes:

- Cada **navegador** tendrá sus propias cookies.
- Las cookies **no diferencian entre usuarios** que utilicen el mismo navegador en una misma sesión del sistema operativo. Muchas veces queda almacenada la información de nuestra tarjeta de crédito al realizar una transferencia bancaria.
- Son **vulnerables a los "sniffer"** (programas que pueden leer el contenido de peticiones y respuestas HTTP) debido a que éstas se realizan en texto plano.
- Las cookies pueden ser **modificadas en el cliente**, lo cual podría aprovechar vulnerabilidades del servidor.

CARACTERÍSTICAS DE LAS COOKIES

Difíciles de gestionar: No podemos leer directamente una cookie, sino que a través de document.cookie obtenermos una cadena de todas las cookies y tenemos que buscar en ella el par nombre=valor que nos interese.

Su número y tamaño está limitado: Se recomienda que los navegadores admitan al menos 20 cookies en cada documento de al menos 4Kbytes cada una, pero esto a veces no es suficiente.

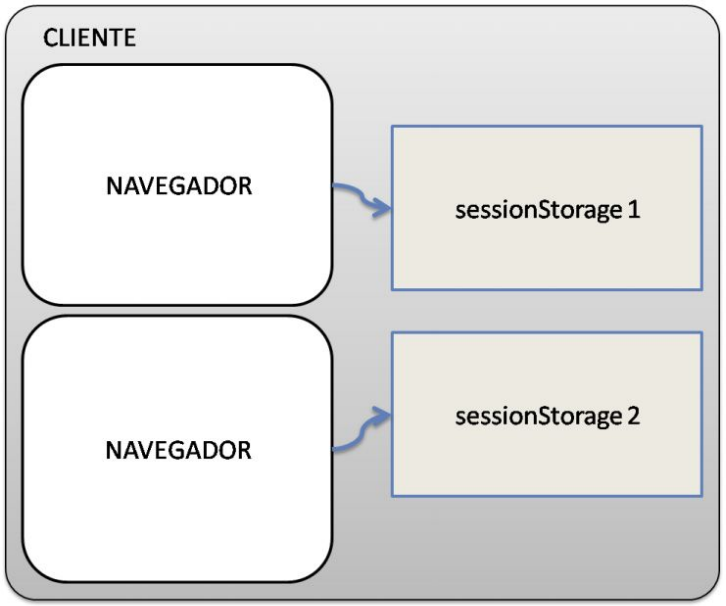
Inseguras: Algunos vectores de ataque se apoyan en el secuestro de cookies para lograr acceso a servicios que requieren acreditación.

WEBSTORAGE. localStorage, sessionStorage VENTAJAS

- La capacidad es de 5Mb frente a los 4Kb de las cookies
- La información no es incluida en cada petición al servidor como sucede con las cookies, sino que solo se obtiene cuando se quiere usar.
- La información es almacenada en pares clave-valor por lo que se puede usar como si se tratase de variables.
- Una página web únicamente puede acceder a la información que ha almacenado ella por lo que es más seguro.

WEB STORAGE
Objeto window.sessionStorage

- El objeto “sessionStorage” se instancia por sesión y ventana, por lo que dos pestañas del navegador abiertas al mismo tiempo y para un mismo sitio Web pueden tener información distinta
- Al cerrar la sesión
  se pierde la información.



Objeto window.localStorage

- Este objeto se extiende a lo largo de múltiples ventanas y múltiples sesiones.
- Puede ser accedido cada vez que se visita el dominio (los subdominios no son válidos) y todas las sesiones abiertas sobre la misma Web ven la misma información

WebStorage. Métodos

- **getItem(nombre):** Devuelve el valor del par identificado por nombre
- **setItem(nombre,valor):** Crea, o modifica si ya existe, el par especificado por nombre y valor.
- **removeItem(nombre):** Borra del almacén el par identificado por nombre.
- **key(indice):** contiene el nombre del par que ocupa la posición índice dentro del almacén.
- **clear():** Borra todos los pares del almacén.

WebStorage. Métodos. Ejemplos

- JavaScript puede crear, leer y borrar del almacenamiento local con los métodos del objeto localStorage
- Almacenamiento de un usuario en almacenamiento local

```
localStorage.setItem ("usuario”,”pepe");
```

- Para leer un parámetro del almacenamiento local en javaScript, lo indicamos así

```
var user=localStorage.getItem("usuario");
```

- Para eliminar utilizamos el método removeItem

```
localStorage.removeItem("usuario");
```

https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

-

WebStorage. StorageEvent

- Cualquier tipo de cambio en el almacén debe disparar un evento de tipo "**StorageEvent**", de forma que cualquier ventana con acceso al almacén pueda responder al mismo
- **Atributos** del evento storage
  - **url** El dominio asociado con el objeto que ha cambiado.
  - **storageArea** Representa el objeto localStorage o sessionStorage afectado.
  - **key** La clave del par clave/valor que ha sido agregado, modificado o borrado.
  - **newValue** El nuevo valor asociado con la clave. Será null si trata de una eliminación.
  - **oldValue** El antiguo valor.


ESCUELA PÚBLICA:
DE TOD@S
PARA TOD@S
Vallecas