

Objetos predefinidos en JS

Web Storage

Módulo: **Desarrollo web en entorno cliente**
CFGS Desarrollo de Aplicaciones Web

¿Qué es una Cookie?

- Una **cookie** es un dato (4kb) almacenado en el lado del cliente.
- Este contiene un conjunto de pares *clave/valor*, como por ejemplo:
["tipoPelícula", "humor"],
["lastpage", 412],...



¿Por qué usar cookies?

- Permitir la personalización específica del usuario basándose en sesiones previas.
- Recuerda las preferencias de los usuarios.
- Mantiene la información de sesión, como por ejemplo una cesta de la compra.
- En general, el almacenamiento de datos que no "pertenecen" a una base de datos.

Deventajas

- Tamaño muy limitado (4 kb por dominio).
- Posibilidad de inconsistencias entre cliente y servidor.



HTML5 Web Storage

- El concepto de HTML5 Web Storage se supone que reemplazará las cookies.
- Se implementa a través de la **API de HTML5 Web Storage**.
- A menudo se hace referencia como "local storage".

HTML5 Web Storage

Las principales ventajas de HTML5 Web Storage:

- (5-10) MB de almacenamiento disponible por dominio.
- Control completo por el cliente, potencialmente se reduce el tráfico de red.

HTML5 Web Storage

¿Cómo determinar si el navegador soporta Web Storage?

```
var webStorageSoportado = ('localStorage' in window) &&  
window['localStorage'] !== null;
```

HTML5 Web Storage

- La API Web Storage se accede a través del objeto **localStorage** (soportado por "todos" los navegadores, incluso IE8)
- Principales métodos:

```
localStorage.setItem(clave, valor);  
var valor = localStorage.getItem(clave);
```


HTML5 Web Storage

Puedes recuperar el número de elementos de **localStorage**:

```
var count = localStorage.length;
```

...y recuperar una clave concreta con su índice:

```
var aKey = localStorage.key(7);
```

HTML5 Web Storage

- Como las claves son strings, pueden almacenar "cualquier" cosa.
- Sin embargo, las claves deben ser únicas.
- Si añadimos un par *clave/valor* en **localStorage** utilizando una clave existente, se sobrescribe la clave existente con el nuevo valor.

HTML5 Web Storage

Puedes acceder a **localStorage** como un array asociativo:

```
// Hacen lo mismo
//
localStorage.setItem(clave, valor);

localStorage[clave] = valor;
```

HTML5 Web Storage

Es necesario saber si una variable del LocalStorage existe antes de usarla.

Esto se logra mediante una condición en javascript:

```
if (localStorage[clave] != undefined) {  
  
    /* la clave si existe */  
  
}
```

HTML5 Web Storage

Se puede borrar una entrada concreta de **localStorage** (para un dominio en cuestión), proporcionando la clave:

```
localStorage.removeItem(clave);
```

HTML5 Web Storage

Para limpiar todas las entradas en **localStorage** (para un dominio concreto):

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

HTML5 Web Storage

¿Cómo obtenemos todas las claves y valores?

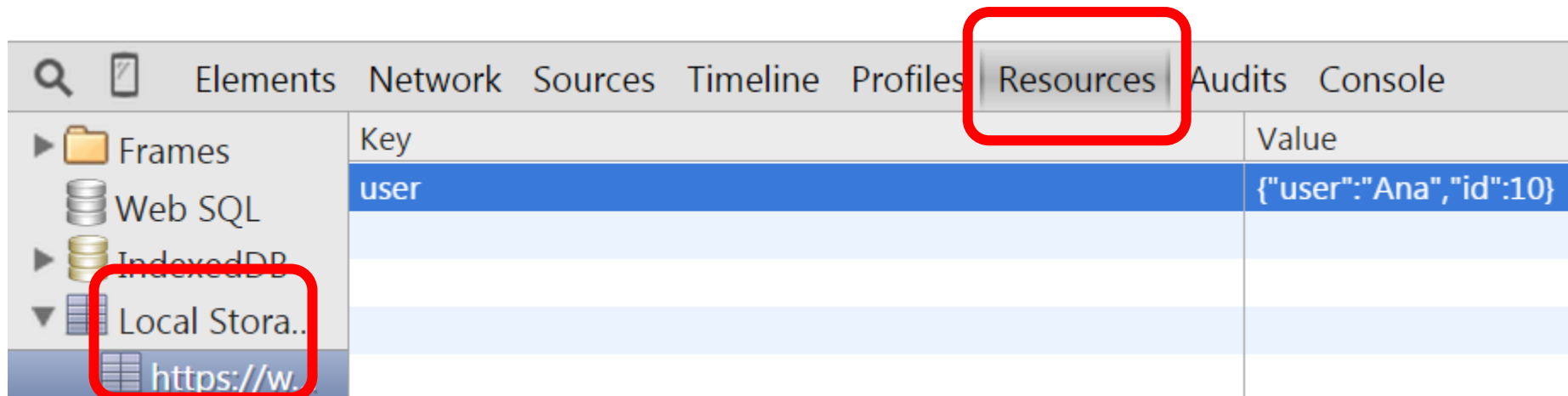
```
for (var i=0; i<localStorage.length; i++) {  
    var clave= localStorage.key(i);  
    var valor = localStorage[clave];  
    alert(valor);  
}
```

Podemos recorrer localStorage cómo si fuera un array asociativo

```
for (var clave in localStorage) {  
    var valor = localStorage[clave];  
}
```

HTML5 Web Storage

Consultar el localStorage en Chrome



HTML5 Web Storage

Ejemplos

```
localStorage.setItem('fechaNacimiento', '1984-07-22');  
  
localStorage.getItem('fechaNacimiento'); // '1984-07-22'  
  
localStorage.length // número de elementos almacenados  
  
localStorage.key(0); // 'fechaNacimiento'  
  
localStorage.removeItem('fechaNacimiento');  
  
localStorage.clear();
```

HTML5 Web Storage

Orden en el que se guardan (clave y valor son Strings)

```
1 > localStorage.setItem("Tenerife", "Teide");  
    < undefined  
2 > localStorage.setItem("Gran Canaria", "Roque Nublo");  
    < undefined  
    > |
```

→ ALFABÉTICO

	Key	Value
▶ Frames		
▶ Web SQL		
▶ IndexedDB		
▼ Local Storage	Gran Canaria	Roque Nublo
	Tenerife	Teide
	epbar::impc	3

HTML5 Web Storage

Utiliza siempre JSON para almacenar objetos y arrays en localSotrage

JSON – JavaScript Object Notation

Formato estándar para representar objetos en pares clave valor como strings

```
var bertObj = { "best_friend": "Ernie" };
```

HTML5 Web Storage

Claves y valores son siempre strings...pero **¿cómo convertir cualquier objeto a string, y viceversa?**

```
var asString = JSON.stringify(asObject);  
var asObject = JSON.parse(asString);
```

Ejemplo

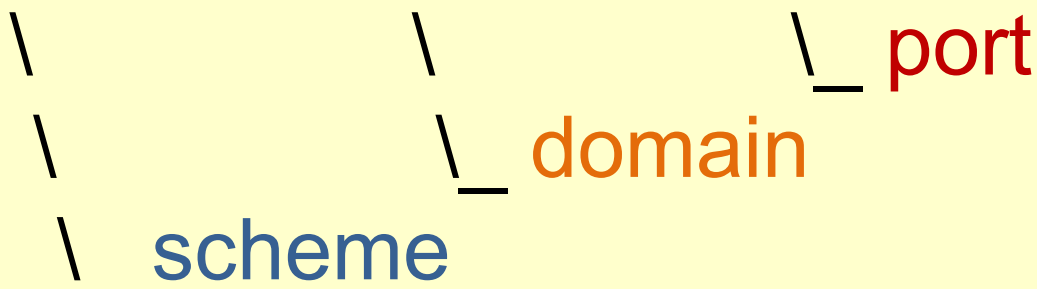
```
localStorage.setItem('user', JSON.stringify({user: 'Ana', id: 10}));  
var user = JSON.parse(localStorage.getItem('user'));
```

```
> A = JSON.stringify({user: 'Ana', id: 10});  
< "{\"user\":\"Ana\",\"id\":10}"  
> B=JSON.parse(A)  
< Object {user: "Ana", id: 10}  
> B.user  
< "Ana"
```

HTML5 Web Storage

Ámbito

http://ejemplo.com:80/



The diagram shows the URL 'http://ejemplo.com:80/' with backslashes (\) placed under each part: 'http', 'ejemplo.com', and '80'. Below these backslashes are labels: '_scheme' (blue) under 'http', '_domain' (orange) under 'ejemplo.com', and '_port' (red) under '80'.

HTML5 Web Storage

- Puesto que el Web Storage está orientado a dominios, los pares de claves/valor de diferentes "aplicaciones" se almacenan en el mismo local storage.
- Por lo que podría ser complicado gestionar ...
 - La claves con nombres duplicados
 - Los esquemas para la generación de claves
 - Ineficiencia a la hora de recuperar claves específicas de una aplicación
 - ...

HTML5 Web Storage

¿Cómo podemos generar claves únicas sin examinar las claves existentes?

Poner como sufijo la hora actual (en milisegundos) al nombre de la clave, por ejemplo: **miClave_1304294652202**.

Código:

```
var curDate = new Date();  
var clave = "miClave_" + curDate.getTime();
```

HTML5 Session Storage

¿Qué pasa si el almacenamiento debe ser orientado a sesiones, es decir, todo debe desaparecer cuando la sesión ha terminado?

- Utiliza **sessionStorage** en vez de **localStorage**
- La API es exactamente la misma...

HTML5 Session Storage

- Parecido a Local Storage
- Dura todo el tiempo que el navegador está abierto
- La apertura de una página en una nueva ventana o pestaña inicia nuevas sesiones

HTML5 Session Storage

```
function incrementLoads() {  
    if (!sessionStorage.miContador) {  
        sessionStorage["miContador"] = 0;  
    }  
    var contador = parseInt(sessionStorage["miContador"]);  
    contador++;  
    sessionStorage["miContador"] = contador;  
    document.getElementById("countDiv").innerHTML = contador;  
}
```