

DÉVELOPPEMENT WEB

CLIENT

Web Storage

API WEB STORAGE

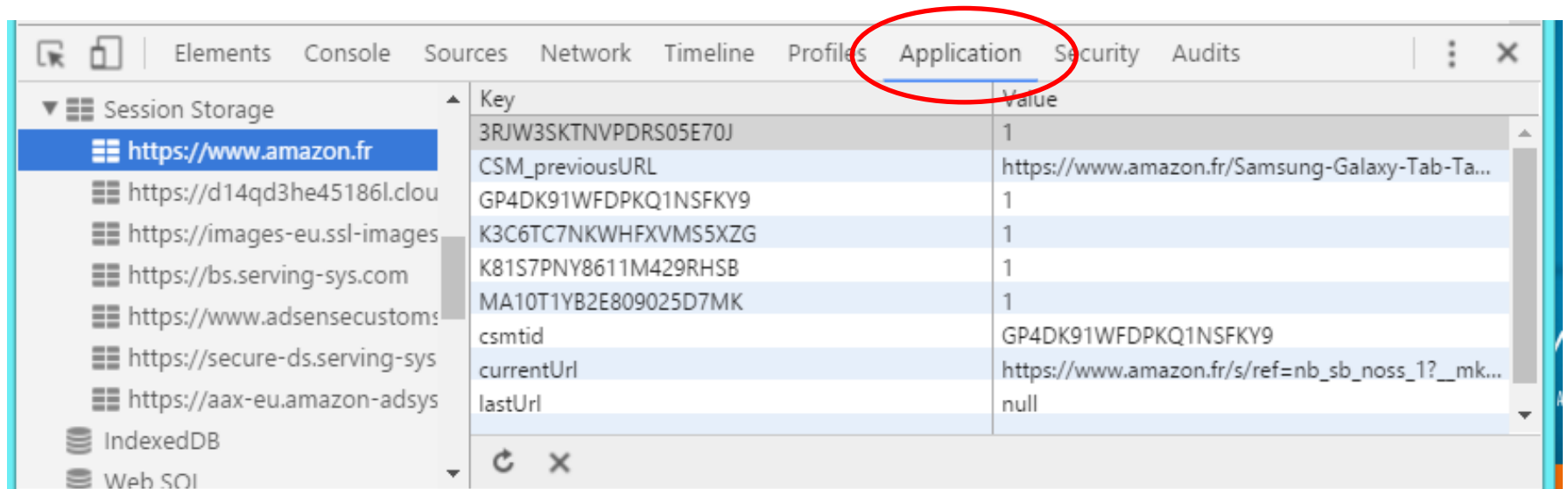
- permet le **stockage de données** dans le navigateur (donc coté client et pas coté serveur)
- technique plus puissante que les cookies car moins limitée en taille (qques kilo pour les cookies, plusieurs Mo pour le webstorage) et consomme moins de ressources réseaux (un cookie est envoyé à chaque requête http sur un domaine)

Attention : les données ne sont pas cryptées et facilement accessibles et modifiable à travers le navigateur.

Attention : certains navigateurs ne rende accessible le webstorage que si la page est chargée à partir d'un serveur

EXEMPLE

- Exemple de web storage utilisé par le site amazon.fr :



The screenshot shows the Chrome DevTools Application tab, which is highlighted with a red circle. The left sidebar shows the 'Session Storage' section for the URL <https://www.amazon.fr>. The main panel displays a table of session storage items.

Key	Value
3RJW3SKTNPDRS05E70J	1
CSM_previousURL	https://www.amazon.fr/Samsung-Galaxy-Tab-Ta...
GP4DK91WFDPKQ1NSFKY9	1
K3C6TC7NKWHFXVMS5XZG	1
K81S7PNY8611M429RHSB	1
MA10T1YB2E809025D7MK	1
csmtid	GP4DK91WFDPKQ1NSFKY9
currentUrl	https://www.amazon.fr/s/ref=nb_sb_noss_1?__mk...
lastUrl	null

API WEB STORAGE

- cette API propose deux modes de gestion (deux classes) :
 - sessionStorage : les données sont conservées tant que le navigateur est ouvert (donc même lors du rechargement de la page)
 - localStorage : les données sont conservées même si on ferme la page
- les données sont stockées sous la forme de couple (clé, valeurs)
 - ➔ la clé et la valeur sont forcément de type chaîne de caractères

API WEB STORAGE

- sessionStorage et localStorage comporte les mêmes méthodes :
- **Ajouter une nouvelle variable** donc un nouveau couple (clé, valeur) [la clé peut être considéré comme le nom de la variable] :
setItem(nom_de_la_clé, valeur_de_la_clé)
- **Récupérer la valeur** d'une variable à partir de la clé :
getItem(nom_de_la_clé)
- **Suppression d'une variable** :
removeItem(nom_de_la_clé)
- **Suppression de toutes les variables** :
clear()

EXEMPLE D'UTILISATION

- une page web gère deux compteurs (sans webstorage)
 - à chaque clic sur le bouton les deux compteurs sont incrémentés

Test webstorage

ajouter 1 Compteur 1 : Compteur 2 :

- après un clic :

Test webstorage

ajouter 1 Compteur 1 : Compteur 2 :

- au rechargement de la page les compteurs reviennent à 0 :

Test webstorage

ajouter 1 Compteur 1 : Compteur 2 :

EXEMPLE D'UTILISATION

- la même page web utilisant du **webstorage** :
 - le compteur 1 est stocké dans localStorage, le compteur 2 dans sessionStorage

- - Après trois clics :

Test webstorage

ajouter 1 Compteur 1 : 3 Compteur 2 : 3

- Après rechargement de la page, les 2 compteurs gardent leur valeur :

Test webstorage

ajouter 1 Compteur 1 : 3 Compteur 2 : 3

- Après fermeture et réouverture de la page, seul compteur1 garde sa valeur :

Test webstorage

ajouter 1 Compteur 1 : 3 Compteur 2 : 0

CODE DE L'EXEMPLE

// déclaration des compteurs

```
var compteur1 = 0, compteur2 = 0;
```

```
window.addEventListener('load',function(){
```

 // récupération du compteur1 dans localStorage s'il existe

```
    if (compteur1 = localStorage.getItem("compteur1")) {
```

```
document.getElementById("compteur1").innerHTML=compteur1;}
```

 // récupération du compteur2 dans sessionStorage s'il existe

```
if (compteur2 = sessionStorage.getItem("compteur2")) {
```

```
document.getElementById("compteur2").innerHTML=compteur2;}
```

 // gestion du clic sur le bouton

```
var cible=document.getElementById("bouton").
```

```
cible.addEventListener('click',compteurClick);
```

```
})
```


CODE DE L'EXEMPLE

```
function compteurClick(e) {  
    // incrémentation des compteurs  
    compteur1 ++;  
    compteur2 ++;  
    // affichage de la nouvelle valeur  
    document.getElementById("compteur1").innerHTML=compteur1;  
    document.getElementById("compteur2").innerHTML=compteur2;  
  
    // sauvegarde dans le webstorage  
    localStorage.setItem("compteur1",compteur1);  
    sessionStorage.setItem("compteur2",compteur2);  
}
```

WEB STORAGE

- La valeur d'une variable du **web storage** est forcément une **chaîne de caractères**
- Pour contourner cette limite et donc stocker tout type d'objet, on va les convertir en une chaîne de caractère au format JSON
 - objet JS -> chaîne de caractères au format JSON
var chaîne = **JSON.stringify**(objet)
 - chaîne de caractères au format JSON -> objet JS
var objet = **JSON.parse**(chaîne)

EXEMPLE

si **mavariabale** est une variable de n'importe quel type donc éventuellement un objet, un tableau ou toute autre structure plus complexe, on utilise alors le **format JSON** :

- stockage dans le sessionStorage :

// transforme mavariabale sous la forme d'un chaîne au format JSON

```
var mavar_json = JSON.stringify(mavariabale);
```

```
sessionStorage.setItem("mavar",mavar_json);
```

- lecture depuis le sessionStorage :

```
var mavar_json = sessionStorage.getItem("mavar");
```

// transforme la chaîne au format JSON en une variable JS

```
var mavariabale = JSON.parse(mavar_json);
```

remarque : fonctionne aussi avec localStorage