**Stockage des données dans le navigateur**

**Web Storage** est une solution adaptée aux besoins actuels de stockage de données variées, dans le navigateur. C'est aussi une technique plus puissante que les cookies, qui sont limités en taille (quelques Ko contre plusieurs Mo pour Web Storage) et qui engendrent un trafic HTTP supplémentaire pour chaque requête (que ce soit pour demander la page web, une image, une feuille de styles, un fichier javascript, etc).

Web Storage met à disposition deux interfaces nommées **sessionStorage** et **localStorage** dont la seule différence concerne la persistance des données. Ces dernières ne sont plus véhiculées sur le réseau HTTP et elles sont facilement accessibles (lecture, modifications et suppression) pour la programmation en JavaScript.



Le stockage de session et le stockage local sont deux espaces distincts.

# Stockage de session sessionStorage

L'interface **sessionStorage** mémorise les données sur la **durée d'une session de navigation**, et sa portée est limitée à la fenêtre ou l'onglet actif. Lors de sa fermeture, les données sont effacées. Contrairement aux cookies,  il n'y a pas d'interférence. Chaque stockage de session est limité à un domaine.

# Stockage local localStorage

L'interface **localStorage** mémorise les données **sans limite de durée de vie**. Contrairement à **sessionStorage**, elles ne sont pas effacées lors de la fermeture d'un onglet ou du navigateur. La portée de **localStorage** est de facto plus large : il est possible de l'exploiter à travers plusieurs onglets ouverts pour le même domaine ou plusieurs fenêtres ; à partir du moment où il s'agit bien sûr du même navigateur.

Il n'y a pas de partage des données Web Storage entre les différents navigateurs qui peuvent être installés sur une même machine.

# Usages et précautions

Le stockage de données dans le navigateur web se prête à différentes applications, particulièrement lorsqu'il s'agit d'exécuter des traitements sans faire intervenir le serveur, ou lorsqu'il faut mémoriser facilement de petites quantités de données pour l'utilisateur pour les faire persister durant sa navigation. Parmi ces avantages :

* stocker rapidement des données en cache sans faire intervenir le serveur (par exemple via AJAX),
* augmenter la performance ressentie et faciliter les développements,
* se passer des cookies et du trafic HTTP supplémentaire qu'ils représentent (un cookie est envoyé à chaque requête effectuée sur un domaine),
* exploiter un espace alloué plus important que la limite imposée par les cookies (fixée à 4 Ko),
* retrouver des données immédiatement à la reconnexion ou les mémoriser pour éviter la perte s'il y a déconnexion.

# Comment y accéder ?

Hormis les spécificités concernant la persistance des données, les méthodes d'accès sont communes :

* **setItem(clé,valeur)** : stocke une paire clé/valeur
* **getItem(clé)** : retourne la valeur associée à une clé
* **removeItem(clé)** : supprime la paire clé/valeur en indiquant le nom de la clé
* **key(index):** retourne la clé stockée à l'index spécifié
* **clear()**: efface toutes les paires

Pour agrémenter le tout, la propriété **.length** renvoie le nombre de paires stockées.

La [**console Javascript**](https://www.alsacreations.com/astuce/lire/1436-console-javascript.html) des navigateurs est un outil essentiel pour tester et vérifier le bon fonctionnement de Web Storage.

**Note** : Les exemples suivants se basent sur sessionStorage mais fonctionneront de la même façon avec localStorage.

# Stockage

sessionStorage.setItem("couleur","vert")

Le premier argument de setItem est la clé (toujours de type String). Elle précise l'endroit où sont stockées les données afin de pouvoir les y retrouver ultérieurement.

# Récupération

var couleur = sessionStorage.getItem("couleur");

Grâce à la clé initialement créée avec setItem il est possible de récupérer facilement les données. Ces dernières sont renvoyées sous la forme d'une chaîne de caractère.

# Suppression

sessionStorage.removeItem("couleur");

Nous supprimons l'élément de session "couleur".

# Suppression totale

sessionStorage.clear();

Suppression complète de toutes les valeurs de session.

# Accès direct

Dans la plupart des situations, les variables seront accessibles directement en tant que membres de l'interface.

sessionStorage.couleur = "vert";

console.log(sessionStorage.couleur);

# Examen et espace alloué

L'espace local alloué, bien que considérablement supérieur à celui disponible via un cookie est cependant limité. Personne ne tient à voir son disque dur submergé de données créées par des milliers de sites consultés. Il faut donc gérer cet espace raisonnablement avec efficacité et parcimonie. Par défaut, Internet Explorer alloue 10 Mo par espace de stockage, et les autres navigateurs (Firefox, Chrome, Opera, Safari) 5 Mo par domaine. En réalité, par mesure de sécurité, on considère l'origine de la page, c'est-à-dire que sont également distingués tous les sous-domaines, ainsi que le port ou le mode de connexion (HTTPS s'il y a lieu). Une exception QUOTA\_EXCEEDED\_ERR est déclenchée si le quota est atteint.

Le navigateur peut aussi très bien choisir de demander l'autorisation à l'utilisateur au préalable avant d'accueillir des données supplémentaires, et d'incrémenter la limite au fur et à mesure.