

# Local et session storage - leaflet

# Stockage de données

Dans le web il est important de pouvoir stocker des données afin de pouvoir les réutiliser entre différentes pages.

Donnez moi un exemple ?



# Stockage de données

Dans le web il est important de pouvoir stocker des données afin de pouvoir les réutiliser entre différentes pages.

Donnez moi un exemple ?

- Un panier
- Un formulaire en cours
- ..etc



# Comment stocker les données

En général on stock les données de deux manières:

- côté serveur via une base de données
- côté client (comme les cookies)

Javascript étant un langage front, nous allons voir la partie client.



# Storage

Nous avons la propriété **localStorage** ou encore **sessionStorage** sur l'objet **window** qui nous permet d'accéder à un objet local Storage.

Cet objet local **storage** dispose de différentes méthodes pour:

- stocker une donnée (clé/valeur)
- récupérer une donnée
- supprimer une donnée...etc



# SessionStorage ou LocalStorage ?

A votre avis quelle pourrait être la différence entre localStorage et le sessionStorage ?



# SessionStorage ou LocalStorage ?

A votre avis quelle pourrait être la différence entre localStorage et le sessionStorage ?

La seule différence est au niveau du stockage et de la durée.

Le sessionStorage **est nettoyé** une fois le navigateur (ou l'onglet) est fermé.  
Le localStorage **lui persiste** malgré tout pour un certain temps.



# 4 méthodes : CRUD

En informatique on parle souvent du CRUD:

- **C**reate
- **R**ead
- **U**ppdate
- **D**eleate

De manière générale une donnée à toujours besoin d'être stockée, d'être lu (récupéré), d'être modifiée et d'être supprimée.

Nous allons voir le CRUD du Storage





# CRUD Storage

Comment créer une donnée ?

Via la méthode **setItem**('key', 'value')

```
localStorage.setItem('monChat', 'Tom');
```



# CRUD Storage

Comment créer une donnée ?

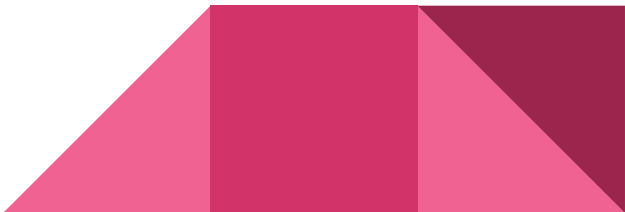
Via la méthode **setItem**('key', 'value')

```
localStorage.setItem('monChat', 'Tom');
```

Comment la modifier ?

Via la même méthode en gardant la clé tout simplement:

```
localStorage.setItem('monChat', 'Jean');
```



# Après la création et la modification, la lecture

Comment lire une donnée ?

Tout simplement via la méthode **get**('key')

```
var cat = localStorage.getItem('monChat');
```



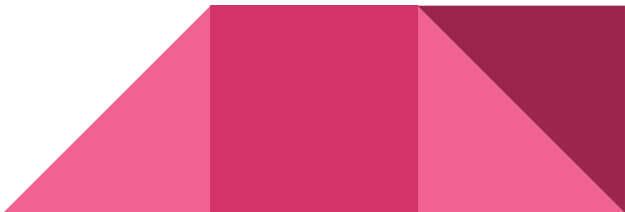
# Pour finir la suppression

Si je veux supprimer un élément, j'ai la méthode **removeItem**('key')

```
localStorage.removeItem('monChat');
```

Et si je veux supprimer **tous** les éléments, j'ai la méthode **clear**()

```
// Effacer tous les éléments  
localStorage.clear();
```



# Doc Storage

## Related Topics

### Web Storage API

#### Storage

##### ▼ Propriétés d'instance

length

##### ▼ Méthodes d'instance

clear()

getItem()

key()

removeItem()

setItem()

# Résumé Storage

En résumé, JS nous fournit un **Web Storage API** qui nous permet de gérer différentes données clé/valeur sur votre site web.

On peut soit:

- Les stocker au niveau du sessionStorage et elles ne durent que la session
- Les stocker au niveau du localStorage elles durent plusieurs sessions



# Third party API

Nous allons utiliser pour notre prochaine TP une Third Party API.

C'est quoi ? Quelle est la différence avec l'API JS ?



# Third party API

Nous allons utiliser pour notre prochaine TP une Third Party API.

C'est quoi ? Quelle est la différence avec l'API JS ?

C'est une API fournie par d'autres organismes (parties) comme Facebook, Google, Twitter..etc qui nous permettent d'utiliser leurs données.

C'est aussi une API d'une librairie tierce (google map, leaflet..etc). On manipule non plus l'API JS mais l'API de l'outil utilisé.

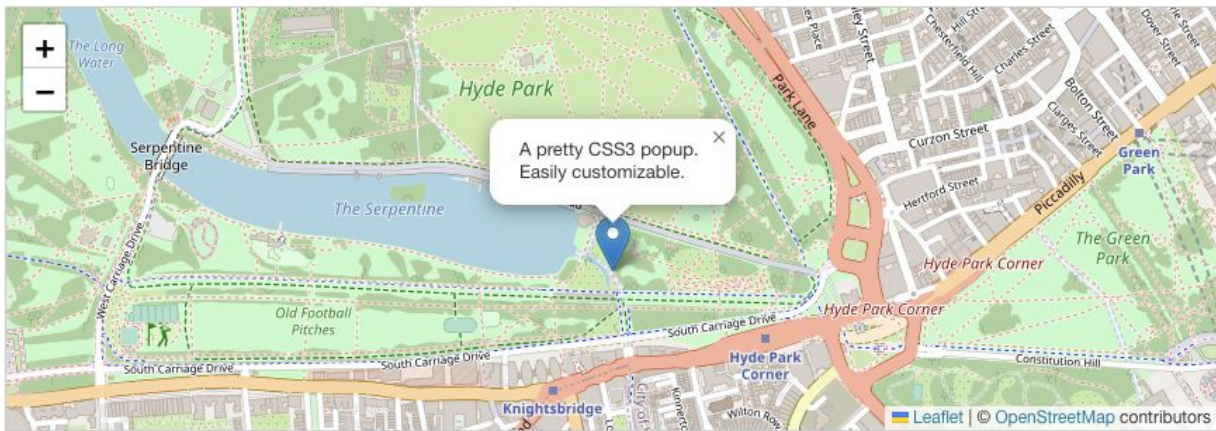




# Leaflet

Allons voir une librairie connue et simple d'utilisation (en plus d'être open source!):

Leaflet: an open-source JavaScript library for mobile-friendly interactive maps



# Comment créer cette carte ?

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contribu
}).addTo(map);

L.marker([51.5, -0.09]).addTo(map)
  .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
  .openPopup();
```

# Commençons par le début

En tant que développeur, on doit souvent aller utiliser des nouveaux outils.  
A votre avis quel est le réflexe à appliquer quand on découvre une nouvelle librairie ou autre ?



# Commençons par le début

En tant que développeur, on doit souvent aller utiliser des nouveaux outils.  
A votre avis quel est le réflexe à appliquer quand on découvre une nouvelle librairie ou autre ?

**Aller voir la doc ! Et non pas tout de suite des tutos..**



## Leaflet API reference

This reference reflects **Leaflet**. Check [this list](#) if you are using a different version of Leaflet.

### Map

[Usage example](#)  
[Creation](#)  
[Options](#)  
[Events](#)

### Map Methods

[Modifying map state](#)  
[Getting map state](#)  
[Layers and controls](#)  
[Conversion methods](#)  
[Other methods](#)

### Map Misc

[Properties](#)  
[Panels](#)

### UI Layers

[Marker](#)  
[DivOverlay](#)  
[Popup](#)  
[Tooltip](#)

### Raster Layers

[TileLayer](#)  
[TileLayer.WMS](#)  
[ImageOverlay](#)  
[VideoOverlay](#)

### Vector Layers

[Path](#)  
[Polyline](#)  
[Polygon](#)  
[Rectangle](#)  
[Circle](#)  
[CircleMarker](#)  
[SVGOverlay](#)  
[SVG](#)  
[Canvas](#)

### Other Layers

[LayerGroup](#)  
[FeatureGroup](#)  
[GeoJSON](#)  
[GridLayer](#)

### Basic Types

[LatLng](#)  
[LatLngBounds](#)  
[Point](#)  
[Bounds](#)  
[Icon](#)  
[DivIcon](#)

### Controls

[Zoom](#)  
[Attribution](#)  
[Layers](#)  
[Scale](#)

### Utility

[Browser](#)  
[Util](#)  
[Transformation](#)  
[LineUtil](#)  
[PolyUtil](#)

### DOM Utility

[DomEvent](#)  
[DomUtil](#)  
[PosAnimation](#)  
[Draggable](#)

### Base Classes

[Class](#)  
[Evented](#)  
[Layer](#)  
[Interactive layer](#)  
[Control](#)  
[Handler](#)  
[Projection](#)  
[CRS](#)  
[Renderer](#)

### Misc

[Event objects](#)  
[global switches](#)  
[noConflict](#)  
[version](#)

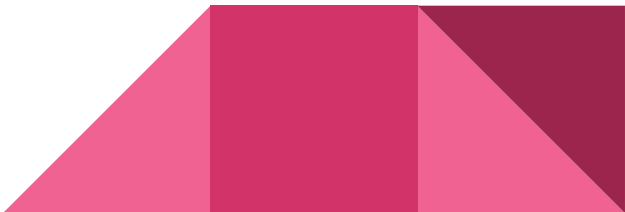
# Un petit coup de pouce

Pour introduire une librairie à notre code, il va falloir importer le js et aussi souvent le css associé.

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css"
integrity="sha256-kLaT2G0SpHechhsozzB+f1nD+zUyjE2LlWPgU04xyI=" crossorigin=""/>
```

```
<script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js"
integrity="sha256-WBkoX0wTeyKc10HuWtc+i2uENFpDZ9YPdf5Hf+D7ewM=" crossorigin=""></script>
```

Maintenant que c'est fait, nous pouvons accéder à leaflet via une variable globale `L`.  
Comme on pouvait faire `window.getItem('nom')` on pourra faire `L.methodeConnue()`



# La librairie est chargée, et maintenant ?

Maintenant on veut créer une carte. Pour cela il nous faut un élément HTML de base assez haut pour contenir la map.

```
<div id="map"></div>
```

```
#map { height: 180px; }
```



# Comment créer une carte ?

Si on suit [la doc](#), nous avons la méthode `map` sur l'objet `L` qui prend en premier paramètre soit un élément HTML soit un ID. Puis une liste d'options

Factory	Description
<code>L.map(&lt;String&gt; id, &lt;Map options&gt; options?)</code>	Instantiates a map object given the DOM ID of a <code>&lt;div&gt;</code> element and optionally an object literal with <code>Map options</code> .
<code>L.map(&lt;HTMLElement&gt; el, &lt;Map options&gt; options?)</code>	Instantiates a map object given an instance of a <code>&lt;div&gt;</code> HTML element and optionally an object literal with <code>Map options</code> .



# Les options ?

La doc indique que le deuxième paramètre est une liste d'options.. Et si on suit la doc on en trouve la liste:

- zoomControl: boolean qui indique si l'utilisateur peut zoomer
- center: indique une position LatLng où centrer la carte
- minZoom: indique le zoom minimum
- ...etc



# La librairie est chargée, et maintenant ?

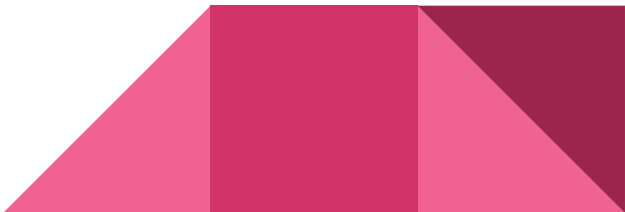
Maintenant on veut créer une carte. Pour cela il nous faut un élément HTML de base assez haut pour contenir la map.

```
<div id="map"></div>
```

```
#map { height: 180px; }
```

Ensuite nous allons utiliser l'API leaflet en JS et créer la carte:

```
var map = L.map('map', {  
  center: [51.505, -0.09],  
  zoom: 13  
});
```



# Leaflet et fond de carte

Attention il faut différencier:

- leaflet qui est une librairie de gestion de carte
- les fonds de cartes disponibles manipulables (les layers).

Nous allons passer par OpenStreetMap (merci encore l'open Source)



# C'est tout ?

Leaflet permet beaucoup de choses:

- on peut créer des markers (avec des popups ou non)
- dessiner des zones sur la carte
- animer des éléments
- ...



# Pourquoi je vous montre cela ?

Parce qu'il est important de savoir se débrouiller avec un nouvel outil.  
Alors on va lancer le TP rapidement



# TP

Le but du jeu est d'afficher Toulouse avec 5 arrêts de chaque ligne de métro.  
Les markers de la ligne A sont rouge et ceux de la ligne B sont jaunes.  
La carte doit être centrée par défaut sur Ap Formation.

—

L'utilisateur dispose d'un formulaire où il peut saisir une longitude et une latitude.  
Au clic sur un bouton, la carte doit se centrer dessus. Mais si l'utilisateur ferme la page et revient, il doit être de nouveau sur cette localisation choisie.

