Aix*Marseille université
Initiative d'excellence
Année 2018-19

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Semaine 2 : TP 6

Javascript et données

Sans serveur web, il est impossible de récupérer directement un fichier de donnée (ce serait une faille de sécurité). La façon la plus simple de récupérer des données en web front est alors de passer par un fichier javascript qui crée une variable contenant les données.

Le fichier `data_annual.js` crée ainsi une variable `raw_data` contenant les données issues de `https://datahub.io/core/global-temp#resource-annual`.

> **A vous** : faites les différents exercices ci-après. Une solution est présentée en fin de chaque partie si vous avez des soucis.

1 Tableaux de données

1.1 données brutes

> **A vous** : Présentez les données dans un paragraphe en utilisant jquery.
>
> **solution** : `donnees_brutes.html`

1.2 une ligne par donnée

Nous allons placer nos données dans une liste. On rappelle qu'en html une liste (non ordonnée) est du type :

```
<ul>
    <li>une ligne de la liste</li>
</ul>
```

Nous allons créer une liste directement avec jquery. Le code suivant crée une liste avec 1 ligne directement avec jquery et l'ajoute dans un `div` d'`id #mon_div`

```
liste = $("<ul>")

ligne = $("<li>")
ligne.text("valeur de la ligne")
liste.append(ligne)

$("#mon_div").html(liste)
```

> **A vous** : Créez une liste contenant les données, une par ligne. On aura besoin de faire des boucle sur le tableau `raw_data`, aidez vous de [itérer dans un `https://www.geeksforgeeks.org/ways-iterating-array-javascript/` pour le faire.
>
> **solution** : `donnees_liste.html`

1.3 préparation des données

Nous (enfin, plutôt, vous) allons formater les données pour les représenter dans un tableau. Nos données sont une extraction brute des données originelle eau format csv. Nous n'avons pas besoin de tout ça.

> **A vous** : a partir des données `raw_data`, créez deux listes la première contenant les années (nommée `label`) et la seconde les moyennes (nommée `data`). POur un indice donnée `i` l'année de la donnée `data[i]`sera `label[i]`.
>
> **solution** : `donnees_tableaux.html`

1.4 dans une table html

A vous : Représentez les données dans une table

solution : `donnees_table.html`

1.5 dans une table datatable

Utilisez le module datatable pour rendre la présentation du tableau jolie. Vous pourrez utiliser le comportement par défaut

solution : `donnees_datatable.html`

2 Graphiques

Utilisez la bibliothèque `http://chartjs.org` pour afficher des données.

La doc `https://www.chartjs.org/docs/latest/` n'est pas très très explicite quant au format de données. Mais `https://tobiasahlin.com/blog/chartjs-charts-to-get-you-started/` permet bien de sy retrouver.

A vous : En utilisant <`https://tobiasahlin.com/blog/chartjs-charts-to-get-you-started/#2-line-chart`, représentez graphiquement la moyenne en fonction de l'année.

solution : `donnees_chartjs.html`

3 Autres données

Testez le module chartjs avec d'autres types de graphique et de données.

Comme il est nécessaire d'avoir un format de donnée précis lorsque l'on veut charger des données avec javascript, on doit souvent les prétraiter. POur cela j'utilise python.

3.1 csv

Le petit programme suivant est celui qui a été utilisé pour transformer un fichier csv en un tableau javascript.

Supposons que l'on veuille regarder les données de `https://datahub.io/core/global-temp#resource-annual` (elles ont été téléchargées là). On commence donc par télécharger les données annuelles : `https://datahub.io/core/global-temp/r/annual.csv` pour obtenir un fichier csv.

On va utiliser python pour le transformer en un fichier directement lisible en javascript

```
import csv

data = []

title = True
for line in csv.reader(open("annual_temp.csv")):
    if title:
        title = False
        data.append(line)
        continue
    data.append([line[0], float(line[1]), float(line[2])])

open("data_annual.js", "w").write('raw_data = ' + str(data))
```

Quelques explications :

— la première ligne qui est les titres est traitée de façon spéciale (pas de conversion)
— tout est chargé sous forme de chaine de caractère car il n'y a pas de délimiteur de chaine (" normalement), il faut donc convertir les données au bon format.
— on a sauvé un fichier js qui affecte une valeur à une tableau de tableau.

A vous : essayer de faire un graphique sur le vent dans marseille pendant les 15 derniers jours : `https://www.meteoblue.com/fr/meteo/archive/export/marseille_france_2995469`

3.2 json

Utiliser le format json avec python est très simple. Une fois le fichier chargé tout se passe comme si l'on manipulait des dictionnaires.

A vous : Essayer de récupérer les données de fermeture de l'action MISCROSOFT des 15 derniers jours. Le fichier json, que j'ai récupéré avec la comande : `https://www.alphavantage.co/query?apikey=demo&function=TIME_SERIES_DAILY_ADJUSTED&symbol=MSFT`

4 Pour aller plus loin

4.1 récupérer directement les données d'internet

On utilise la méthode ajax pour récupéer les données au format csv, puis un plugin jquery pour lire les fichiers csv. Ceci nous permet de traiter, sans python, directement les données depuis internet.

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Magie jquery</title>

        <link href="https://fonts.googleapis.com/css?family=Indie+Flower" rel="stylesheet">

        <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-csv/1.0.3/jquery.csv.min.js"></sc
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.bundle.js"></script
        <script>
            $(function() {
                $.ajax({
                    url: "https://datahub.io/core/global-temp/r/annual.csv",
                    type: "GET",
                    success: function (result) {

                        tab = $.csv.toArrays(result,
                                             {onParseValue: $.csv.hooks.castToScalar});
                        data = []
                        labels = []
                        first = true
                        tab.forEach(function(element) {
                            if (first) {
                                first = false;
                                return
                            }
```

```
                    data.push(element[2])
                    labels.push(element[1])
                });
                var ctx = $('#myChart');
                new Chart(ctx, {
                            type: 'line',
                            data: {labels: labels,
                                   datasets: [{data: data,
                                               label: "temp",
                                               fill:false
                                              }]
                                  },
                });
            }
        });
    });
    </script>
</head>
<body>
<h1>Un graphique du réchauffement climatique</h1>
<canvas id="myChart" width="400" height="400"></canvas>
</body>
</html>
```

On voit que le nombre d'appels augmente grandement. La méthode jquery utilisée pour récupérer les données est $.ajax. Elle est asynchrone : on envoie une requête à un serveur et, lorsqu'il répond, on exécute la fonction définie dans success avec comme paramètre la réponse du serveur.

4.2 serveur python

Si l'on veut afficher directement ses propres données, il faut fabriquer un serveur web. C'est heureusement très facile pour un ca d'utilisation aussi simple.

On créer ici un serveur python minimal dont le but est de servir les fichiers d'un dossier choisi à l'avance. Il doit être ouvert aux requêtes extérieurs (puisque il sera appelé depuis notre autre fichirr html), on supprime donc les restrictions d'appels CORS.

On aura besoin d'un serveur web python. On a choisi flask :

— `pip3 install flask` le serveur

— `pip3 install flask-cors` la gestion des cors

Puis on crée le serveur proprement dit :

```
from flask import Flask, request, send_from_directory
from flask_cors import CORS

# set the project root directory as the static folder, you can set others.
app = Flask(__name__, static_url_path='/')
CORS(app)

@app.route('/<path:filename>')
def send_file(filename):
    return send_from_directory('./', filename)
```

```
if __name__ == "__main__":
    app.run(port=8080)
```

Ce programme lance un serveur web qui répond au routes `/nom` où `nom` est un fichier du répertoire où est le fichier python et il rend ce fichier. Pour que notre serveur puisse donner ses fichier à n'importe qui on autorise les requêtes d'autres serveurs avec la commande `CORS(app)`.

Une fois le serveur lancé :-avec la commande `python3 mon_server.py` nous avons un serveur web qui écoute à l'adresse : `http://localhost:8080`. Il rend les fichier qui sont dans son répertoire. Par exmple si l'on tape `http://localhost:8080/annual_temp.csv` on obtiendra le fichier annual_temp.csv s'il est dans le dossier du fichier python.

En remplaçant l'appel ajax par `http://localhost:8080/annual_temp.csv` on peut maintenant directment récuperer nos données.