université angers
FACULTÉ DES SCIENCES
Unité de formation et de recherche

L3 Informatique - 2024/2025
UE Développement Web
David Lesaint

TP JS 4 - Graphiques

Décompressez l'archive déposée sur Moodle pour ce TP. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. Pensez à consulter le site MDN.

On propose une page web intégrant des graphiques (voir Figure 1) de données météorologiques et administratives fournies par l'ODRE et l'INSEE. L'implémentation `JS` s'appuie sur les promesses et la librairie `Plot`, elle-même fondée sur la librairie D3.
La page permet au visiteur de visualiser

- les températures moyennes quotidiennes de 4 départements sur la période 2018-2023,
- les températures minimales, maximales et moyennes des mois de 2022 dans le Maine-et-Loire,
- visualiser la division administrative d'une région française en départements et communes.

Ce démonstrateur vous aidera à visualiser ce qui est attendu.

Le dossier décompressé contient les éléments suivants :

- **data** : dossier des fichiers de données au format `CSV`, `JSON` et `XML`.
- **graphiques.html** : le fichier `HTML` de la page web.
- **graphiques.css** : la feuille de styles.
- **graphiques.js** : le fichier principal `JS` produisant les graphiques.
- **communes.js** : le module d'extraction des communes.
- **communes.php** : le script `PHP` paginant la liste des communes.
- **départements.js** : le module d'extraction des départements.
- **régions.js** : le module d'extraction des régions.
- **températures_départementales.js** : le module d'extraction des températures départementales.
- **températures_anjou.js** : le module d'extraction des températures angevines.
- **utils.js** : le module d'utilitaires.

Le code qui vous est fourni est commenté et partiellement fonctionnel. Vous devrez remplacer ou compléter différents éléments des fichiers `JS` pour répondre aux questions.

Exercice 1. Utilitaires [CSV, Fetch]
Le fichier **utils.js** contient deux fonctions utilitaires :

- `parseCSV` pour parser le contenu d'un fichier `CSV`,
- `p_fetch` pour requêter par `URL` un fichier au format `CSV`, `JSON` ou `XML`.

1. Implémentez la fonction `parseCSV` en vous appuyant sur la méthode de chaînes `split`.

Exercice 2. Régions et départements [XML, opérations asynchrones séquentielles/parallèles]

1. Le fichier **régions.csv** codifie les régions françaises (voir description des champs). Le fichier **régions.js** implémente la promesse `p_régions` pour en récupérer le contenu et le transformer en un tableau d'objets. Implémentez la promesse `p_régions` par chaînage de promesses en vous aidant des utilitaires importés.

2. Le fichier **départements.xml** codifie les départements français (voir description des balises). Le fichier **départements.js** implémente la promesse `p_départements` pour en récupérer le contenu et le transformer en un objet. La transformation requiert les régions extraites par `p_régions`. Implémentez la promesse `p_départements` en parallélisant l'extraction des régions et la récupération du fichier `XML`. Utilisez la méthode DOMParser.parseFromString pour analyser le contenu XML.

Exercice 3. Températures [JSON, jq, opérations asynchrones parallèles]

1. Le fichier **49_59_65_83.json** fournit les températures minimales, moyennes et maximales relevées quotidiennement pour les départements 49, 59, 65 et 83 sur la période 2018-2023. Ce fichier a été obtenu en transformant le fichier complet des relevés français (renommé **temperatures.json**) à l'aide du programme jq. Téléchargez `jq` pour votre architecture (vérifiez avec `uname -a`) et l'ajoutez à votre `PATH`. Testez la commande ci-dessous et exercez-vous à produire d'autres types de filtrage.

```
cat temperatures.json | \
    jq '[ .[] | {tmoy:.fields.tmoy, tmin:.fields.tmin, tmax:.fields.tmax,
       date:.fields.date_obs, code:.fields.code_insee_departement}] | \
    sort_by(.date, .code) | \
    map(select(.code | match("(49)|(59)|(65)|(83)")))' \
    > mon_49_59_65_83.json
```

2. Le fichier **températures_départementales.js** implémente la promesse `p_températures_départementales` pour récupérer le contenu du fichier `JSON` et le transformer en un objet. Le fichier **températures_anjou.js** implémente la promesse `p_températures_anjou` pour calculer les températures minimales, moyennes et maximales mensuelles en 2022 dans le département 49 à partir des relevés fournis par `p_températures_départementales`. La réponse de `p_températures_anjou` (si elle est tenue) est un tableau d'objets. Implémentez cette promesse en consommant et en transformant la réponse de `p_températures_départementales`.

Exercice 4. Communes [Curl/Postman, pagination de résultats]
Le script `PHP` **communes.php** fournit la liste des communes d'une région en paginant les résultats par blocs successifs de 100 communes.
Le script répond aux requêtes `HTTP POST` en acceptant deux paramètres :

— `région` : le code de la région demandée

— `bloc` : le numéro du bloc de communes demandé.

Il renvoie alors le bloc de communes sous la forme d'un tableau d'objets `JSON` où chaque objet a 3 propriétés :

— `nom` : le nom de la commune (chaîne)

— `département` : le code du département de la commmune (chaîne)

— `région` : le code de la région (chaîne).

Il renvoie un tableau vide si le bloc demandé est vide.

1. Testez le script `PHP` avec différents paramètres en utilisant la commande `curl`, p. ex. :

- `région=52, bloc=3` : communes de la région PDL allant de Viellevigne (44) à La Ménitré (49).
- `région=52, bloc=13` : communes allant de Saint-Gilles-Croix-de-Vie (85) à La Faute/Mer (85).
- `région=52, bloc=14` : tableau vide.

Voici un exemple de ligne de commande (adaptez l'URL) :

```
curl \
    -d 'région=1&bloc=1' \
    -H 'Content-Type: application/x-www-form-urlencoded' \
    -X POST \
    'http://localhost:8080/l3dw/tp-js-4/corrections/communes.php'
```

Alternativement, installez l'application graphique `Postman` pour testez le script.

2. Le fichier **communes.js** implémente la fonction `extraireCommunes(communes, région, bloc)`. Elle ajoute au tableau `communes` de la région `région` toutes les communes figurant dans les blocs de rang $\geq$ `bloc`. Les blocs (tableaux) sont simplement concaténés sans transformation. La fonction est récursive et requête le script `PHP` à chaque appel en renvoyant la promesse `Fetch` correspondante. Réimplémentez la fonction `extraireCommunes`.

Le fichier **communes.js** implémente aussi la fonction `p_communes`. Cette fonction prend en paramètre le code d'une région (chaîne) et renvoie une promesse se résolvant en un tableau d'objets représentant toutes les communes de la région. La promesse parallélise au préalable l'extraction des départements (promesse `p_départements`), celle des régions (promesse `p_régions`) et celle des communes de la région en appelant `extraireCommunes`. La transformation opérée consiste à nommer départements et régions qui ne sont que codifiés par `extraireCommunes`.

Exercice 5. Graphiques [Plot]

1. Le fichier **graphiques.js** produit les graphiques de la page en utilisant la librairie Plot. Il consomme la promesse `p_températures_départementales` pour générer le premier graphique de températures et la promesse `p_températures_anjou` pour le second. Implémentez l'utilisation de la promesse `p_températures_anjou` en vous inspirant de l'implémentation fournie pour `p_températures_départementales` et en suivant les commentaires pour paramétrer le graphique des températures angevines.

2. Le fichier **graphiques.js** implémente un écouteur sur le menu déroulant permettant de sélectionner une région. Le cas échéant, la fonction de rappel consomme la promesse renvoyée par `p_communes` pour générer l'arborescence des communes de la région. Réimplementez la fonction de rappel fournie à la méthode `p_communes(région).then` en suivant les commentaires.

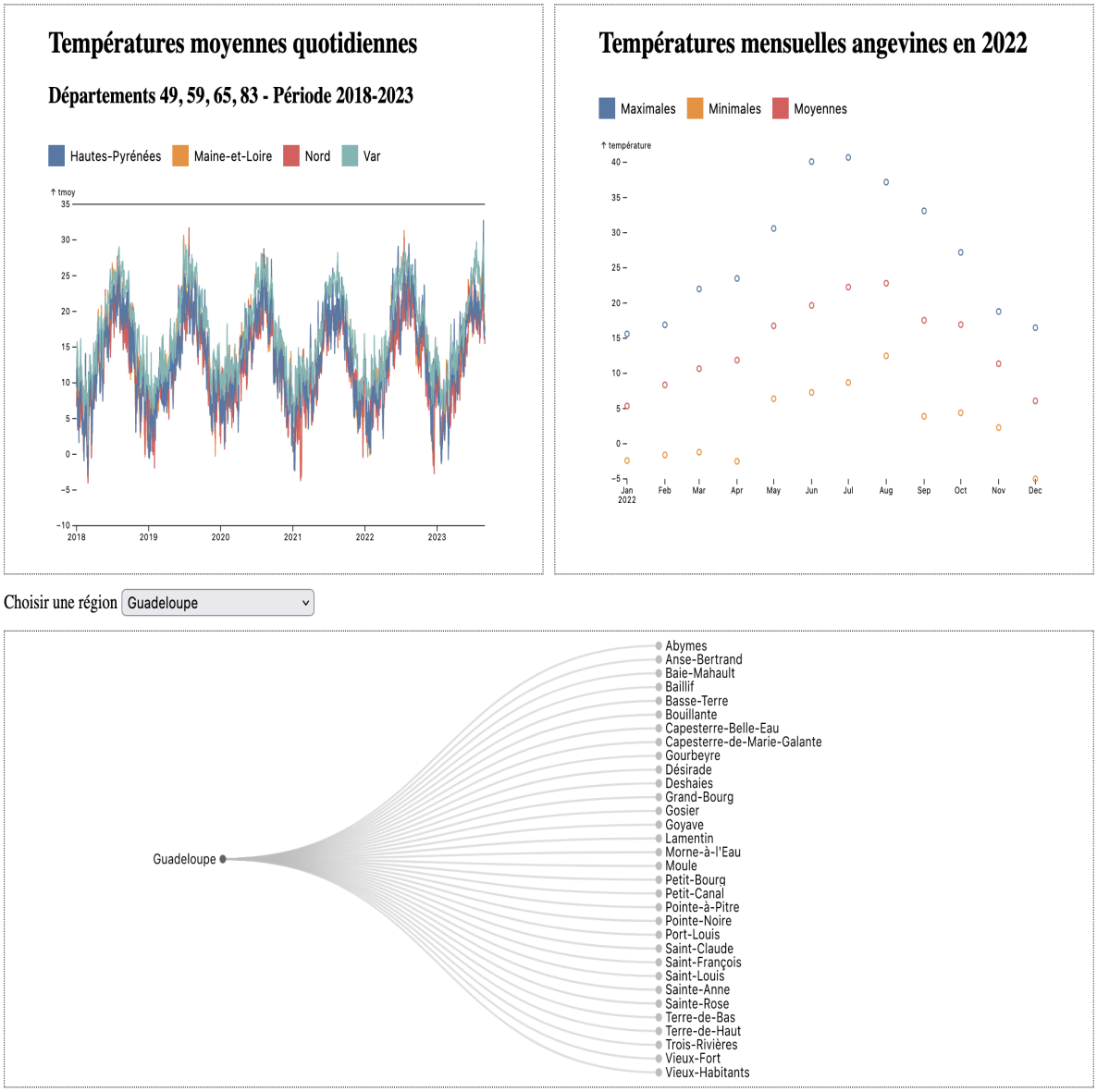

FIGURE 1 – Graphiques de températures et de divisions administratives