Vue – Exercice 5

Utilisation d'une API délivrant des données - Ajax

I. Présentation

Téléchargez le site fourni sur Moodle.

A partir du site fourni, vous devrez créer une page qui affiche les informations relative à un artiste, puis les dates et lieux de ses prochains concerts. Puis vous créerez un moteur de recherche permettant d'interroger une base de données concernant des artistes.

II. Quelques éléments de culture

A. Utilisation d'une API

Notre application va utiliser des données qu'elle récupérera via une adresse web (une URL[1]).

Pour visualiser ces données, allez à l'adresse :

https://rest.bandsintown.com/artists/angele?app_id=db2d9ab207b873e3336d985c9f200b54

Les données affichées sont présentée au format JSON, c'est le format que nous avons pris l'habitude d'utiliser pour décrire les données que nous manipulons avec VueJS.

Une partie des échanges de données entre ordinateurs sont réalisés suivant ce format.

B. JSON

JavaScript Object Notation est une manière de décrire une donnée structurée.

Le texte JSON présenté à droite décrit deux personnes pour lesquelles nous possédons trois informations.

nom et *prenom* sont des informations de type texte, *anneeNaiss* est une donnée de type numérique.

La structure de cette information la rend similaire à une table de base de données :

nom	prenom	anneeNaiss
"Patitucci"	"John"	1959
"Page"	"Jimmy"	1944

```
[
   {
        "nom"          : "Patitucci",
        "prenom"    : "John",
        "anneeNaiss":1959
   },
   {
        "nom"          : "Page",
        "prenom"    : "Jimmy",
        "anneeNaiss":1944
   }
]
```

Exemple de texte json

Les symboles *[* et *]* délimitent un **tableau** : un ensemble d'éléments possédant une structure similaire (ou une sémantique équivalente). Ici, le tableau contient deux éléments similaires : deux personnes.

En JSON, la syntaxe générale d'un tableau est : *[élément1 , élément2, ... , dernierElement]*

Les symboles *{* et *}* délimitent un **objet**. Un objet regroupe généralement plusieurs champs.

En JSON, la syntaxe générale d'un objet est :

```
{
    "nom du champ 1" : "valeur du champ 1",
    "nom du champ 2" : "valeur du champ 2",
    …
    "nom du dernier champ " : "valeur du dernier champ"
}
```

Les valeurs textuelles sont entourées par des guillemets ce qui n'est pas le cas des valeurs numériques,

1 Unified Ressource Locator : adresse d'un document sur le web

ni des valeurs booléennes (true/false).

Remarque : Un champ peut contenir un objet ou un tableau, un tableau peut contenir des objets ...

Exercice :

- Allez ici pour voir le résultat d'une requête soumise à l'API d'open data soft
- Combien de champs l'objet principal comporte-t-il ?
- Nommez ces champs et indiquez leur nature (texte, nombre ou objet)
- Le champ *records* contient-il un objet ou un tableau ?
- Dans les résultats obtenus, quel est le nom du second enregistrement ?

C. Ajax

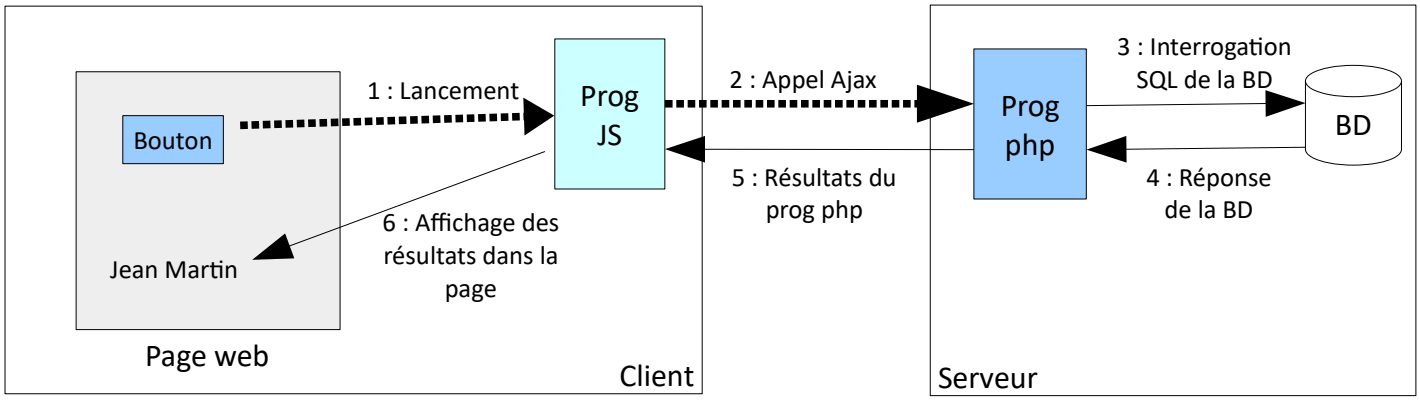
Lorsque vous utilisez *Deezer*, *Facebook* ou *Twitter*, vous constatez qu'une petite partie de la page web est modifiée sans que le navigateur ait besoin de recharger l'ensemble de la page.
Ajax (Asynchronous Javascript And Xml) est utilisé dans ce but.
Ajax regroupe un ensemble de techniques qui permettent à un programme javascript de modifier seulement une partie d'une page web en y incorporant des données (reçues au format XML ou JSON).
Sans Ajax, pour modifier le contenu d'une page web, le navigateur client doit recharger toute la page.
Avec Ajax, clients et serveurs peuvent communiquer de manière plus souple, un navigateur peut ainsi demander à ce que soit modifiés seulement certains éléments de la page web.

Dans nos programmes Javascript, les gestionnaires d'événements pourront récupérer des résultats générés par des programmes php exécutés par le serveur.

Le dialogue asynchrone (indépendant du chargement de la page) entre le client et le serveur web est rendu possible par l'objet Javascript *XMLHttpRequest*. Cet objet étant complexe à utiliser, des programmeurs ont conçu des bibliothèques de programmes Javascript permettant d'en simplifier la manipulation. Nous utiliserons essentiellement la fonction *get* définie dans la bibliothèque *Axios.*

D. Architecture mise en jeu

Ici, notre page web contiendra un bouton permettant au programme javascript de lancer l'exécution d'un appel Ajax, ce qui déclenchera l'exécution d'un programme php sur le serveur. Le programme php ira chercher des informations dans une base de données. Ces informations seront récupérées par le programme js comme résultat de l'appel Ajax.



E. API utilisée

Consultez la documentation de l'API à cette adresse .

Vous aurez besoin de la clé d'API pour effectuer des simulations : db2d9ab207b873e3336d985c9f200b54

III. Manipulation

Dans le fichier *index.html*, on trouve déjà les liens vers les feuille de styles, le fichier *vue.js* et le fichier *progAppli.js*.

Dans le programme javascript on a indiqué que l'application s'exécutera dans l'élément #appli3.

A. Affichage d'un artiste

Modifiez l'appel ajax pour qu'il fournisse la donnée disponible à cette adresse.

Notez que la partie *then* de l'appel ajax place la donnée reçu dans la variable *artiste* de notre application.

Modifiez la page web afin qu'elle affiche le nom de l'artiste.

Affichez également l'image de l'artiste ainsi que son nombre d'abonnés.

On veut que le nombre d'abonnés soit affiché avec un séparateur permettant de mieux le lire (par exemple 1422390 est plus lisible sous la forme 1 422 390). Dans l'application, repérez la fonction capable d'effectuer ce formatage. Côté html, faites appel à cette fonction pour traiter et afficher le nombre d'abonnés.

Affichez le nombre de concerts prévus et le lien vers la page Facebook de l'artiste.

Si un artiste n'a aucun concert prévu, on préfère afficher " Aucun concert prévu " plutôt que ou " 0 concerts prévus ". Dans la rubrique "methods", créez une fonction *texteNbConcerts* qui traite le nombre de concerts et qui renvoie le texte adéquat. Utilisez cette fonction pour l'affichage du nombre de concerts.

B. Test

Modifiez l'URL de l'appel ajax de façon à accéder aux données d'un autre artiste. Vérifiez que le programme fonctionne toujours.

C. Liste des concerts

On veut établir la liste des prochains concerts de l'artiste.

Dans la documentation de l'API, cherchez à quelle adresse on peut trouver ces données.

Dans la fonction *lancerRecherche*, dupliquez la fonction *axios.get* effectuant l'appel ajax de façon à récupérer ces nouvelles données. Vous placerez les données reçues dans *data.concerts.*

Affichez la liste des concerts dans la page.

D. Recherche

On veut afficher les données concernant un artiste dont le nom est saisi dans le formulaire se trouvant en haut de la page.

Cherchez à relier le champ de saisie du formulaire à la donnée *nom_artiste_recherche*.

Lorsque l'on soumet le formulaire, il faut relancer la fonction *lancerRecherche*, en empêchant le rechargement de la page web ...