TP3 - Programmation client (fetch, DOM, événements)

Polytech Nantes - Info 3

Objectifs du TP

Dans le TP1 vous avez vu comment créer et mettre en forme une page HTML statique. Dans le TP2 vous avez fait vos premiers programmes JavaScript en manipulant des données au format JSON.

Dans ce TP3, nous vous proposons de **programmer** (en JavaScript) **la partie client d'une application web** permettant de parcourir des données musicales (genres, artistes, albums). Vous verrez en particulier les notions suivantes :

- **Récupération de données** auprès d'un serveur via la méthode fetch. La requête étant effectuée au moyen du protocole HTTP.
- **Gestion de l'asynchronisme** (ex : appel à `fetch()`) via le mécanisme des promesses et async / await de JavaScript.
- **Modification et création d'éléments HTML** en JavaScript via les méthodes du DOM.
- **Réponse aux interactions** de l'utilisateur avec certains éléments de votre page web via l'utilisation d'écouteurs d'évènement.
- **Animations** (simples) via des transitions CSS.

Outils

Pour réaliser ce TP vous aurez besoin de :

- Un éditeur de texte au choix,
- Votre navigateur préféré.

Nous allons également utiliser trois outils pour simplifier le développement :

- **npm** (node package manager) nous permettra d'installer les outils de validation de votre code ainsi qu'un outil permettant de recharger votre page HTML à chaque fichier modifié.
- **gulp** va permettre d'automatiser la réalisation des tâches décrites ci-dessus à chaque modification des fichiers HTML, CSS et JavaScript.
- **json-server** fournira la partie serveur de ce TP. Ce module crée une API REST (on verra ça en cours bientôt) à partir d'un fichier JSON lui servant de base de données. Vous pourrez ainsi récupérer des données en faisant une requête HTTP sans avoir à écrire le code du serveur (que nous réaliserons dans le TP4).

Installation

Vous trouverez sur Madoc une archive contenant les données et scripts nécessaires à la réalisation de ce TP. Téléchargez cette archive et extrayez son contenu.

- A la racine de votre projet (là où est stocké votre fichier `package.json`) lancez l'installation de tous les outils en tapant la commande suivante dans un terminal : `npm install`
- Maintenant que les outils sont installés vous pouvez lancer gulp. On va le lancer via une commande npm : `npm test`.
- Cette commande devrait lancer votre navigateur par défaut (si ce n'est pas le cas, ouvrez l'adresse http://localhost:3002/ dans votre navigateur) et commencer à surveiller vos fichiers HTML, CSS et JavaScript (comme aux TP1 et TP2).
- Pour développer la partie client de votre application, nous devons également lancer (**dans une autre console**) la partie serveur via la commande : `npm start`.
- Si le serveur est bien lancé, vous devez pouvoir consulter l'API à l'adresse suivante : http://localhost:3000/.

Note : selon votre installation et votre configuration de système, `localhost` peut correspondre à une résolution de nom d'hôte vers adresse IP différente (IPv4 : dans le préfixe `127.0.0.0/8`, IPv6 : adresse standardisée `::1/128`) ; le service doit être le même quelle que soit la version de protocole Internet utilisée (IPv4 ou IPv6), mais selon la configuration de votre système et / ou de votre client web, il utilise préférentiellement l'une ou l'autre des versions du protocole Internet. Vous pouvez toujours *forcer* l'utilisation de l'une ou l'autre en spécifiant directement une adresse IP dans l'URL, par exemple `http://[::1]:3000` (IPv6) ou `http://127.0.0.1:3000/` (IPv4).

Partie 1 - Genres et artistes

Préliminaires

- Ajoutez le code HTML nécessaire dans `index.html` afin que votre script main.js soit chargé le plus rapidement possible, mais exécuté une fois le HTML interprété par votre navigateur (indice : c'est une histoire d'attribut…)
- Ajoutez un `console.log()` (avec le message de votre choix) à votre fichier `main.js` et vérifiez dans la console JavaScript de votre navigateur que tout fonctionne comme prévu.

Remplissage du sélecteur de genre musical `<select>` (Promise)

Tout le code créé dans cette partie devra être ajouté dans une fonction `loadGenres()` que vous appellerez à la fin de votre script `main.js`.

- Ouvrez votre navigateur à l'adresse suivante : http://127.0.0.1:3000/genres et vérifier que la liste (JSON) des genres s'affiche bien.
- Utilisez la méthode `fetch()` de l'API JavaScript afin de récupérer la liste des genres. Dans un premier temps, nous afficherons cette liste dans la console.
  - Attention, `fetch()` renvoie une promesse. Il vous faudra donc gérer deux cas :
    - La promesse est tenue : il faut traiter la réponse obtenue avec la méthode `then()` (dont le callback aura un argument que vous nommerez `response`).
    - La promesse n'est pas tenue : il faut traiter l'erreur avec la méthode `catch()`.
  - Même si la promesse est tenue, il est possible qu'une erreur ait pu avoir lieu. Vous pouvez vérifier ceci avec la propriété `response.ok`.
  - La réponse obtenue étant un fichier JSON, nous devons parser les données à l'aide de la méthode `response.json()`. Attention, cette méthode renvoie également une promesse !
- Maintenant que vous avez des données, il va falloir utiliser le DOM afin de remplir l'élément `<select>` avec chacun des genres récupérés. Nous allons créer un élément `<option>` pour

chacun des genres. Le texte de l'élément contiendra le nom du genre et la valeur de l'élément contiendra l'id du genre. Quelques indications utiles :

- Vous avez deux possibilités pour récupérer un / des élément(s) HTML
  * `document.querySelector()` vous permet de sélectionner le premier élément correspondant à un sélecteur CSS passé en paramètre. Par exemple `document.querySelector('body p');` renverra le premier élément `<p>` du `<body>`. Si on veut obtenir tous les `<p>` il faut utiliser `document.querySelectorAll()`
  * `document.evaluate()` permet de récupérer tous les éléments correspondant à l'évaluation d'une expression XPath. Par exemple `document.evaluate('/html/body//p', document, null, XPathResult.ANY_TYPE, null);` renverra un itérateur permettant l'accès à tous les paragraphes HTML du corps du document,
- `document.createElement()` permet de… créer un élément HTML
- L'élément `<select>` possède une méthode `add()` permettant de lui ajouter un élément `<option>` (qu'il faudra créer auparavant).

Prise en compte du genre sélectionné

- Créez une fonction `genreChanged()` qui pour l'instant ne fait qu'afficher un message dans la console.
- Dans votre fonction `loadGenres()`, Utilisez la méthode `addEventListener()` de votre élément `<select>` afin que votre fonction `genreChanged()` soit appelée lorsque l'utilisateur sélectionne une option (évènement `'change'`).
- Modifiez la fonction `genreChanged()` afin d'afficher dans la console l'option sélectionnée dans le `<select>`.
- Notre fonction `genreChanged()` étant très courte nous allons la supprimer et la remplacer par une fonction fléchée `(evt) => { // code de votre fonction fléchée }` directement dans l'appel à `addEventListener()`. Vous pourrez faire de même dans tous les appels à des fonctions anonymes de votre code.

Remplissage de la liste des artistes (async / await)

Il s'agit maintenant d'utiliser la syntaxe async / await plutôt que la création et la résolution de promesses. Vous verrez, c'est plus simple et lisible. Pour la suite des exercices de ce TP, seule la syntaxe `async + await` sera conservée.

Maintenant que la liste de genres est remplie et que nous savons exécuter du code lorsque l'utilisateur modifie sa sélection dans la liste des genres nous allons créer une fonction `loadArtists( genre )`. Cette fonction devra être appelée par l'écouteur d'évènement créé dans la partie précédente (sélection d'un nouveau genre), mais aussi dès la fin du chargement des genres. Elle nous permettra de :

- Mettre à jour le titre `<h2>` situé dans la `<section id="#main">`. Par exemple, si le genre "rock" est sélectionné on affichera "Top rock artists" en modifiant la propriété `textContent` du titre.
- Remplir le paragraphe `<p>` situé juste après le titre `<h2>` avec le descriptif du genre sélectionné.
  - Vous devez passer à `loadArtists` l'ensemble des données JSON correspondant au genre sélectionné.

  - À partir du tableau `genres` reçu qui contient la liste des genres, construisez un dictionnaire pour faciliter la mise en correspondance entre l'id du genre sélectionné et sa description.
- Remplir la liste `<ul>` située après le paragraphe avec le nom (`<h3>`) et la photo (`<img>`) du groupe / artiste (le `<h3>` sera inséré dans un lien `<a>` afin de rendre celui-ci cliquable un peu plus tard dans ce TP). Pour réaliser ceci vous aurez besoin de :
  - Faire une nouvelle requête `fetch()` à l'adresse suivante :
  http://127.0.0.1:3000/genres/**nom_du_genre**/artists afin de récupérer la liste des artistes correspondant au genre sélectionné. Vous remplacerez bien entendu `__nom_du_genre__` par le genre sélectionné (ex : rock).
  - Utiliser la méthode forEach sur le tableau d'artistes que vous aurez récupéré via votre appel à `fetch()` pour parcourir tous les artistes
  - Utiliser (entre autres) les méthodes du DOM (querySelector, createElement, setAttribute, appendChild) pour créer et ajouter les éléments HTML nécessaires.

Il vous faudra également compléter légèrement le fichier `style.css` afin d'obtenir un **résultat proche** de celui-ci.

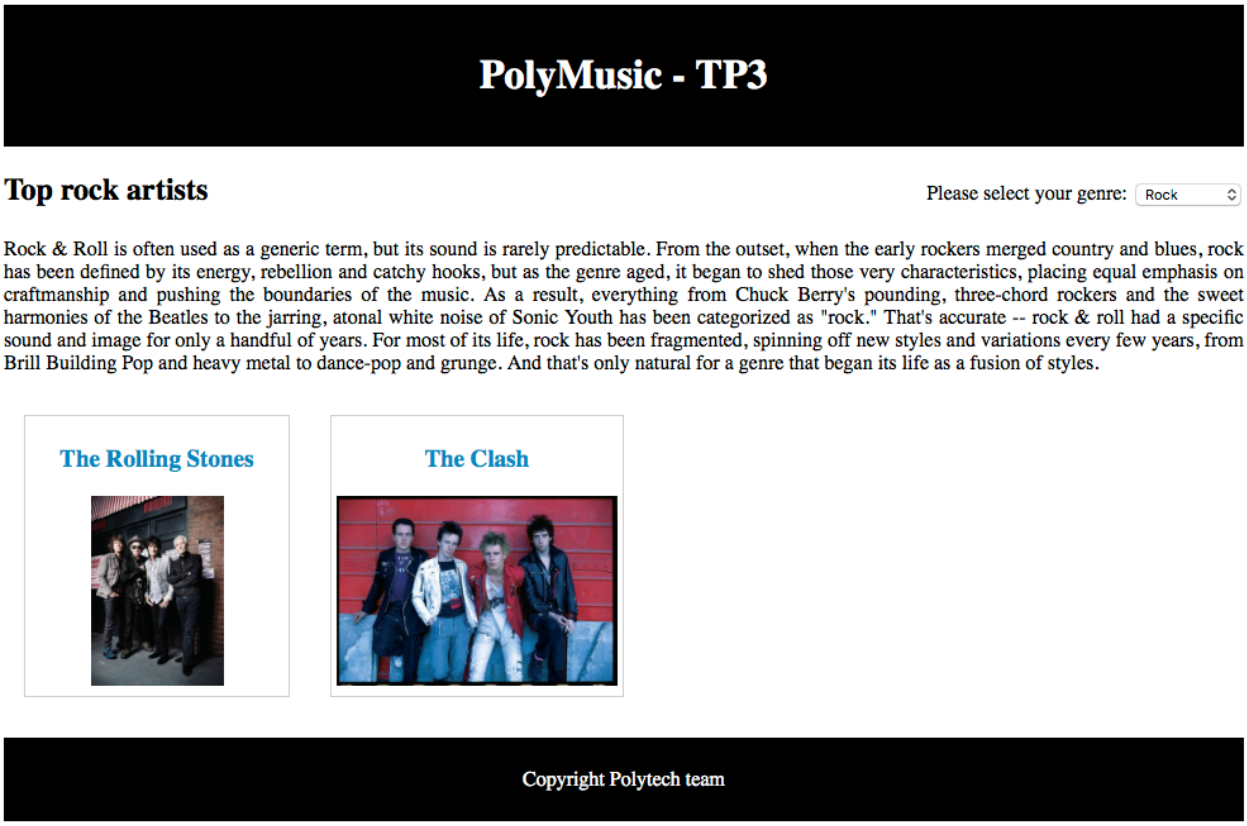

Figure 1: Liste des artistes

Partie 2 - Albums

Dans cette deuxième partie, nous allons ajouter une fenêtre popup qui apparaitra lorsque l'utilisateur cliquera sur le nom d'un artiste. Cette fenêtre fera apparaitre la liste de ses albums.

En réalité, cette fenêtre existe déjà (c'est l'`<aside>` présent dans `index.html`), mais est caché grâce à une règle CSS. Il va donc falloir la faire apparaitre.

Pour effectuer ces tâches, vous créerez une fonction `artistSelected( evt )` qui sera appelée lors d'un clic sur le nom d'un artiste. Cela implique l'utilisation de `addEventListener()` dans votre fonction `loadArtists()`.

Apparition et centrage du popup

- Avant de faire apparaitre la fenêtre popup, il va falloir faire une nouvelle requête `fetch()` afin de récupérer les albums correspondant à l'artiste sélectionné.
  - L'adresse à utiliser est : http://127.0.0.1:3000/artists/**id_de_l_artiste**/albums. Vous remplacerez bien sûr `__id_de_l_artiste__` par l'identifiant de l'artiste dont le nom a été cliqué. Ceci peut être fait en :
    * Modifiant votre fonction `loadArtists()` afin d'insérer l'identifiant de chaque artiste en tant qu'attribut HTML `id` du lien `<a>` associé à chaque nom d'artiste.
    * Récupérant cet `id` via la propriété `evt.target.parentElement.id` de l'évènement reçu en paramètre de `artistSelected( evt )` (car lorsque l'on clique sur le lien `<a>`, c'est bien le titre `<h3>` situé à l'intérieur qui est la cible de l'évènement).
- Une fois les données récupérées sans erreurs, nous allons pouvoir afficher le popup. Pour cela il faut récupérer cet élément à l'aide de la fonction DOM `querySelector()`. Vous modifierez ensuite les propriétés de style (CSS) de cet élément :
  - `style.visibility` permettra de rendre l'élément visible (pas vraiment en fait, car son opacité est toujours à 0)
  - `style.opacity` permettra de passer l'opacité de ce popup à 1 (pour qu'il soit réellement visible).
  - `style.transition` permettra d'ajouter une animation CSS pour rendre le changement d'opacité graduel (voir ici pour plus de détails).
  - `style.top` et `style.left` permettront centrer cette fenêtre. En effet, la taille du popup variera en fonction du nombre d'albums que nous insérerons dans celui-ci, nous devons donc ajuster sa position pour qu'il soit toujours centré. Pour ce centrage, les propriétés `clientWidth` et `clientHeight` du `<body>` et du popup `<aside>` vous seront utiles.
  - Veuillez noter qu'il est également possible (et plus propre) de créer des règles CSS pour afficher / cacher le popup et d'utiliser uniquement du Javascript pour activer ces règles (avec une classe par exemple). Dans ce cas, vous pourrez centrer votre popup avec `top` et `left`, mais aussi la propriété `transform` (et des translations négatives).
- Il vous faudra également ajouter un écouteur d'évènement afin de faire disparaitre (graduellement) le popup lorsqu'on clique sur le bouton "Ok".

Remplissage de la liste des albums

Vous devriez maintenant pouvoir vous débrouiller seuls pour ce dernier remplissage d'éléments HTML à partir des données JSON récupérées à partir du serveur. Vous devrez également ajuster le fichier CSS afin d'obtenir un **résultat proche** de l'image ci-dessous.

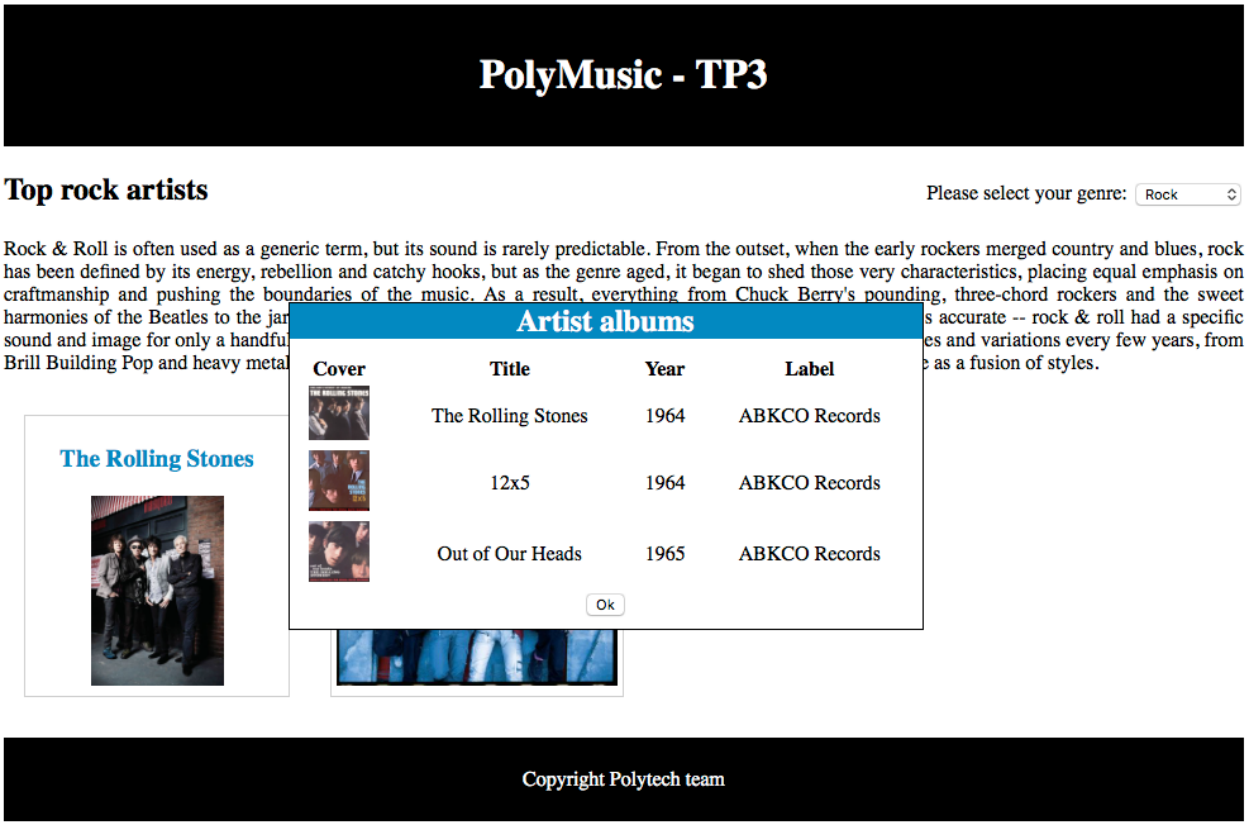

Figure 2: Liste des albums

A rendre…

Vous rendrez sur Madoc une archive contenant votre code (HTML, CSS, Javascript) supposé fonctionnel (`npm start`) et testable (`npm test`).

L'évaluation du TP sera faite sur deux critères :

2. Quantité de fonctionnalités mises en place et respect des consignes.
3. Code HTML et CSS valide, pas d'erreur au linter Javascript (modulation de la note jusqu'à -50%).