

Techniques d'intégration multimédia

Programmation spécialisée

582-504-SF | GR0001 et GR0002 | SEPTEMBRE 2021

TP2 – Programmation asynchrone

Pondération : 25% de la session

Écorcé du projet

L'objectif de ce travail est de vous familiariser avec la **programmation asynchrone** et la documentation au format **Markdown**. Tous les styles visuels (CSS) vous seront fournis et aucun balisage HTML n'est requis.

Pour réaliser ce projet, vous devrez réaliser **5 exercices** à partir du dossier remis par l'enseignant. Aucune librairie externe ne doit être utilisée pour effectuer les requêtes AJAX.

#	Titre de l'exercice	Pointage
EX1	Markdown	2 points
EX2	Conversion d'un <code>then()</code> en <code>await</code>	2 points
EX3	Horloge numérique	4 points
EX4	Recherche d'universités canadiennes	4 points
EX5	Liste aléatoire d'utilisateurs	6 points

Exercice 1 : Markdown

Reproduire en Markdown le document HTML affiché dans la capture d'écran ci-dessous. Respectez les niveaux sémantiques des éléments du document (**h1** à **h6**, **table**, **p**, **strong**, **em**, **a** et **code**). Un fichier de texte brut est fourni ([src/EX1_texte-brut.txt](#)). Convertir ce texte brut en Markdown dans le fichier **EX1_markdown.md**.

TP2 - Programmation asynchrone `<h1>`

- Titre du cours: Programmation spécialisée `, , `
- Programme: Techniques d'intégration multimédia
- Session: Automne 2021
- Pondération: 25% de la session
- Nombre de questions: 5 questions

Sommaire des exercices `<h2>`

#	Nom	Pointage	<code><table></code>
EX1	Markdown	2 points	
EX2	Conversion d'un <code>then()</code> en <code>await</code>	2 points	<code><code></code>
EX3	Horloge numérique	4 points	
EX4	Recherche d'universités canadiennes	4 points	
EX5	Liste aléatoire d'utilisateurs	6 points	

Exercice 1: Markdown `<h3>`

Reproduire en markdown une capture d'écran d'une page Web.

Outil pour visualiser du markdown: markdownlivepreview.com ``

Exercice 2: Conversion d'un `then()` en `await` `<h3>, <code>`

Convertir avec un `await` un script JavaScript asynchrone (`Promise`) qui utilise un `then()`. `<code>, `

Exercice 3: Horloge numérique `<h3>`

Création d'une horloge numérique qui affiche les heures, les minutes et les secondes.

Exercice 4: Recherche d'universités canadiennes `<h3>`

Création d'une zone de recherche d'universités canadiennes avec l'aide d'un API.

Exercice 5: Liste aléatoire d'utilisateurs `<h3>`

Afficher une liste d'utilisateurs à l'aide d'un API.

Exercice 2 : Conversion d'un `then()` en `await`

OBJECTIF

Convertir avec la syntaxe `await` un script JavaScript (TypeScript) asynchrone (`Promise`) qui utilise la syntaxe `then()`. Vous devrez également utiliser un `try... catch()` pour prévenir les erreurs provenant de la requête AJAX.

NOTES

- Ne pas oublier de faire un `try... catch()`.
- Conserver la méthode `fetch()` déjà en place pour faire votre requête AJAX.
- Le script doit fonctionner de la même façon qu'auparavant lorsqu'aucune conversion n'était effectuée.

INSTRUCTIONS

1. Ouvrez avec *Live Server* le fichier `EX3_conversion-then.html` et observez les résultats obtenus dans la console de débogage.
2. Ouvrez le fichier `EX2_conversion-then.ts` et analysez le code dans l'état actuel.
3. Lire attentivement la fonction `obtenirUniversiteUSA()`.
4. Convertisez le `then()` de la fonction `obtenirUniversiteUSA()` avec la syntaxe `await` et gérer les erreurs avec un `try... catch()`.

Exercice 3 : Horloge numérique

OBJECTIF

Produire un script JavaScript (TypeScript) qui affichera les heures, les minutes et les secondes courantes. L'affichage du temps courant doit être rafraîchi chaque seconde.

NOTES

- Utiliser l'objet JavaScript `Date` pour afficher l'horloge.
 - Format de temps attendu : `<HH>:<MM>:<SS>`
 - `<HH>` : Heures avec deux chiffres
 - `<MM>` : Minutes avec deux chiffres
 - `<SS>` : Secondes avec deux chiffres
 - Exemple : 08:06 :09
 - Afficher la date courante avec le local `fr-CA`
 - Exemple : Lundi 3 mai 2021
 - Permettre à l'utilisateur de modifier le local de la date courante pour `en-CA`
 - Utiliser une liste déroulante (dropdown)

```
<select name="localDate">
    <option value="fr-CA">Français</option>
    <option value="en-CA">English</option>
</select>
```
 - Ne pas oublier de modifier l'attribut `lang` sur la balise `html` avec la valeur du local courant.
- Implémenter un mode lumineux et un mode sombre.
 - Le mode lumineux est actif de 5 heures à 20 heures
 - Le mode sombre est actif de 20 heures à 5 heures
- Utiliser le `window.setInterval()` pour rafraîchir le temps courant toutes les secondes.
- L'affichage du temps est généré en JavaScript (TypeScript) et injecté dans la page [HTML EX3_horloge-numérique.html](#).
- Aucun code TypeScript n'est fourni pour cet exercice. Vous êtes responsable de programmer l'entité de la fonctionnalité. Vous devez personnaliser les styles graphiques de l'interface (CSS).
 - La composante doit être « responsive », c'est-à-dire qu'elle doit bien s'afficher dans tous les types d'appareils (mobile, tablette, ordinateur, etc.).

Mode lumineux

09:20:35

Lundi 3 mai 2021

Mode sombre

20:02:03

Lundi 3 mai 2021

Exercice 4 : Recherche d'universités canadiennes

OBJECTIF

Produire un script JavaScript (TypeScript) qui permettra de rechercher par nom d'université les universités canadiennes. Pour aller récupérer la liste d'universités, vous devrez utiliser l'api [University Domains and Names Data List](#). Le balisage HTML du formulaire est fourni dans le fichier `EX4_rechercher-universites.html`. Vous devriez utiliser ce balisage pour injecter le nombre de résultats dans la balise `p.nombre-resultats` et injecter la liste de résultats dans la balise `ul.resultat-recherche`.

NOTES

- La requête AJAX doit être effectuer avec l'objet `XMLHttpRequest`.
- L'URL de l'api : <http://universities.hipolabs.com/search>.
 - Paramètres de recherche: `?name=<nomUniversite>&country=canada`.
 - Exemple d'URL complète :
<http://universities.hipolabs.com/search?name=laval&country=canada>
 - Prendre connaissance de la structure du JSON retourné à l'URL :
<http://universities.hipolabs.com/search>.
- La recherche doit être lancée sur ces deux événements :
 - L'événement '`submit`' du formulaire
 - L'événement '`input`' du champ de recherche
- La liste de résultats est une liste de liens qui ouvrira le site Web de l'université sélectionné dans un nouvel onglet. Les libellés de ces liens doivent correspondre aux noms des universités.

INSTRUCTIONS

1. Ouvrez avec *Live Server* le fichier `EX4_rechercher-universites.html` et observez ce qui est affiché dans votre navigateur. Analysez le balisage HTML et repérez où vous injecterez en JavaScript les résultats de recherche.
2. Ouvrez le fichier `EX4_rechercher-universites.ts` et analysez le code dans l'état actuel. Observez les constantes et les deux événements qui vous sont fournis.
3. Complétez la fonction `rechercherUniversitesCanadiennes()`. Dans cette fonction, effectuez votre requête HTTP avec `new XMLHttpRequest()`. Avec le résultat de cette requête, construirez la liste de résultats et affichez le nombre de résultats obtenus.
4. Regarder les captures d'écran des résultats attendus à la page suivante.

Recherche d'universités canadiennes

Qué

Rechercher

12 résultats

- [Institut National de la Recherche Scientifique, Université du Québec](#)
- [École de technologie supérieure, Université du Québec](#)
- [École nationale d'administration publique, Université du Québec](#)
- [Institut Armand-Frappier, Université du Québec](#)
- [Université du Québec à Trois-Rivières](#)
- [University of Québec](#)
- [Université du Québec en Outaouais](#)
- [Université du Québec en Abitibi-Témiscamingue](#)
- [Université du Québec à Montréal](#)
- [Université du Québec à Chicoutimi](#)
- [Université du Québec à Rimouski](#)
- [Télé-université, Université du Québec](#)

Recherche d'universités canadiennes

Lav

Rechercher

1 résultat

- [Université Laval](#)

Recherche d'universités canadiennes

xx

Rechercher

Aucun résultat

Exercice 5 : Liste aléatoire d'utilisateurs

OBJECTIF

Produire un script JavaScript (TypeScript) qui affichera une liste aléatoire de dix utilisateurs à l'aide de l'api <https://randomuser.me/>. Cette liste doit afficher la photo de profil des dix utilisateurs. Lorsque l'événement '`click`' est déclenché sur l'image, la fiche correspondant à l'utilisateur sélectionné s'affichera. Cette fiche exposera le prénom et le nom, la photo de profil, l'adresse, la ville, l'état et le code postal de l'utilisateur sélectionné.

NOTES

- L'URL de l'api : <https://randomuser.me/api/>.
 - Paramètre pour obtenir 10 utilisateurs : `?results=10`
 - Exemple d'URL complète : <https://randomuser.me/api?results=10/>
 - Prendre connaissance de la structure du JSON retourné à l'URL : <https://randomuser.me/api/>.
- Dans la `section#liste`, vous devez ajouter des boutons contenant une vignette d'utilisateur. L'attribut `alt` de l'image doit être '`Afficher la fiche de ${user.name.first}`'
 - Au clic sur un bouton, la fiche correspondante doit apparaître dans la `section#fiche`.
- Une fonction asynchrone `obtenirListeAleatoireUtilisateurs()` doit être définie. Elle contient une promesse que vous devez gérer avec un `await` et un `try... catch()`.
- La solution doit être développée en JavaScript natif (TypeScript) avec la méthode `fetch()`. Aucune bibliothèque n'est permise.
- Lire les commentaires dans le HTML et respecter le balisage HTML attendu.
- Aucun code TypeScript n'est fourni pour cet exercice. Vous êtes responsable de programmer l'entité de la fonctionnalité.

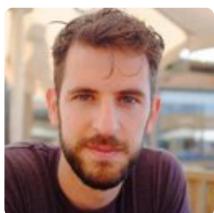
Liste aléatoire d'utilisateurs



Liste aléatoire d'utilisateurs



Jacob Fortin



- 7471 Balmoral St
- Sidney
- Yukon
- J3U 2R8