Cahier d'exercice - Réutilisation des Web Components cross-frameworks

> Ce document n'est qu'un premier brouillon en attente d'une relecture complète et de tests en situation réelle. Merci de me signaler toute erreur, incohérence ou manque de clarté.

Exercice 1 : Créer et exporter un Web Component réutilisable avec modern-web / open-wc

Objectif

- Créer un Web Component réutilisable qui affiche la température actuelle d'une ville donnée, en utilisant l'API **Open-Meteo** (https://open-meteo.com/en/docs/meteofrance-api).
- Le composant doit accepter un attribut `city` pour spécifier la ville.
- Le composant doit être exporté en tant que module JavaScript, afin de pouvoir l'utiliser facilement dans différents projets.

Instructions

1. **Initialiser le projet :**
   - Créez un projet de bibliothèque de composants avec `open-wc` ou `modern-web` en suivant les instructions d'installation et de configuration (comme expliqué dans l'exercice précédent).
2. **Créer le composant `weather-widget` :**
   - Le composant doit accepter un attribut `city` qui détermine la ville dont on veut afficher la température.
   - Faites une requête à l'API Open-Meteo pour récupérer la température en fonction de la ville.
   - Le composant doit afficher la température de la ville.
3. **Exporter le composant :**
   - Exportez le composant sous forme de module JavaScript, afin qu'il puisse être facilement importé dans d'autres projets.
4. **Test du composant :**
   - Créez un fichier HTML minimaliste pour tester le Web Component en l'intégrant dans la page et en spécifiant une ville dans l'attribut `city`.
5. **Publication du composant (optionnel) :**
   - Si vous le souhaitez, vous pouvez préparer votre composant pour le rendre disponible via un CDN ou pour le publier sur un registre npm.

Question

- Pourquoi est-il important d'exporter le composant en tant que module JavaScript ? Qu'est-ce que cela apporte en termes de réutilisabilité dans différents projets ?

Exercice 2 : Créer et exporter un Web Component réutilisable avec StencilJS

Objectif

- Créer un Web Component réutilisable qui affiche la température actuelle d'une ville donnée, en utilisant l'API **Open-Meteo** (https://open-meteo.com/en/docs/meteofrance-api).
- Le composant doit accepter un attribut `city` pour spécifier la ville.
- Le composant doit être exporté en tant que module JavaScript, afin de pouvoir l'utiliser facilement dans différents projets.

Instructions

1. **Initialiser le projet StencilJS** :
   - Créez un projet StencilJS en exécutant la commande :

     ```
     npm init stencil
     ```

2. **Créer le composant `weather-widget`** :
   - Dans le fichier `weather-widget.tsx`, créez un Web Component qui accepte l'attribut `city` et fait une requête à l'API Open-Meteo pour récupérer la température.
3. **Exporter le composant** :
   - Exportez le composant `weather-widget` en tant que module JavaScript.
4. **Test du composant** :
   - Testez le composant dans un fichier HTML minimaliste en utilisant la commande suivante.
5. **Publication du composant (optionnel)** :
   - Préparez le composant pour qu'il puisse être publié sur npm ou mis à disposition via un CDN.

Question

- Quelles sont les différences principales entre un Web Component créé avec StencilJS et un Web Component créé avec `open-wc` ? Dans quel cas préférez-vous l'un par rapport à l'autre ?

Exercice 2 : Créer et exporter un Web Component réutilisable avec StencilJS

Objectif

- Créer un Web Component réutilisable qui affiche la température actuelle d'une ville donnée, en utilisant l'API **Open-Meteo** (https://open-meteo.com/en/docs/meteofrance-api).
- Le composant doit accepter un attribut `city` pour spécifier la ville.
- Le composant doit être exporté en tant que module JavaScript, afin de pouvoir l'utiliser facilement dans différents projets.

Instructions

1. **Initialiser le projet StencilJS** :
   - Créez un projet StencilJS en exécutant la commande :

     ```
     npm init stencil
     ```

2. **Créer le composant `weather-widget`** :
   - Dans le fichier `weather-widget.tsx`, créez un Web Component qui accepte l'attribut `city` et fait une requête à l'API Open-Meteo pour récupérer la température.
3. **Exporter le composant** :
   - Exportez le composant `weather-widget` en tant que module JavaScript.
4. **Test du composant** :
   - Testez le composant dans un fichier HTML minimaliste en utilisant la commande suivante.
5. **Publication du composant (optionnel)** :
   - Préparez le composant pour qu'il puisse être publié sur npm ou mis à disposition via un CDN.

Question

- Quelles sont les différences principales entre un Web Component créé avec StencilJS et un Web Component créé avec `open-wc` ? Dans quel cas préférez-vous l'un par rapport à l'autre ?

Exercice 3 : Ajouter des tests unitaires (open-wc)

Objectif

- Ajouter des tests unitaires à votre Web Component `weather-widget` créé dans l'exercice 1

- Vérifier que le composant récupère et affiche correctement la température pour une ville donnée en utilisant les outils de testing fournis par `@open-wc/testing`.

Instructions

1. Installez `@open-wc/testing` dans votre projet.
2. Créez un fichier de test `weather-widget.test.js` pour tester le comportement du composant.
3. Dans le test, vérifiez que :
   - Le composant affiche le texte "Loading..." avant que les données ne soient récupérées.
   - Après la récupération des données, le composant affiche la température.
4. Lancez les tests pour vérifier leur bon fonctionnement.

Question

- Pourquoi est-il important d'écrire des tests unitaires pour vos composants ? Quelles sont les bonnes pratiques en matière de tests pour les Web Components ?

Exercice 4 : Documentation

Objectif

- Ajouter de la documentation à votre Web Component en utilisant TSDoc.
- Générer une documentation développeur pour le Web Component qui décrit ses propriétés, méthodes et événements.

Instructions

1. Ajoutez des commentaires TSDoc au-dessus de chaque propriété, méthode et événement dans votre composant `weather-widget`.
2. Utilisez un générateur de documentation comme `TypeDoc` ou un outil similaire pour générer la documentation en HTML ou PDF.
3. Testez le rendu de la documentation générée pour vous assurer qu'elle est claire et complète.

Question

- Pourquoi est-il important de bien documenter ses composants ? Comment une bonne documentation peut-elle améliorer la collaboration dans une équipe de développement ?

Exercice 5 : StencilJS

Objectif

- Reproduire les étapes des exercices précédents (création d'un Web Component, ajout de tests unitaires, documentation) mais cette fois-ci en utilisant StencilJS.
- Le Web Component doit être réutilisable, bien documenté et testé, tout comme dans les exercices précédents.

Instructions

1. **Initialiser un projet StencilJS** en utilisant `npm init stencil`.
2. **Créer le composant `weather-widget`** avec StencilJS, en suivant la même logique que dans les exercices précédents (affichage de la température via l'API Open-Meteo).
3. **Ajouter des tests unitaires** pour vérifier que le composant fonctionne correctement avec l'API.
4. **Documenter le composant** avec TSDoc et générer la documentation développeur.
5. **Exporter le composant** pour qu'il soit réutilisable dans d'autres projets.
6. **Tester et valider le composant** dans un fichier HTML minimaliste.

Question

- Quelles sont les différences entre StencilJS et `open-wc` en termes de fonctionnalités et de configuration ? Pourquoi choisir l'un ou l'autre ?

Exercice 6 : Angular

Objectif

- Créer un projet Angular minimaliste, dans lequel vous allez intégrer le Web Component `weather-widget` créé dans les exercices précédents.
- Utiliser les outils de StencilJS et Angular pour intégrer un Web Component dans un projet Angular.

Instructions

1. **Initialiser un projet Angular** :
   - Créez un projet Angular minimaliste en suivant la documentation officielle.
     - Commande : `ng new weather-app`
2. **Configurer Angular pour accepter les Web Components** :
   - Déclarez `CUSTOM_ELEMENTS_SCHEMA` dans le module Angular.
   - Assurez-vous que le Web Component peut être utilisé dans les templates Angular.
3. **Intégrer le Web Component `weather-widget` dans Angular** :
   - Importez le fichier `weather-widget.js` dans votre projet Angular.
   - Utilisez le Web Component dans le template de l'application Angular, en passant un attribut `city` pour spécifier la ville.

4. **Test du projet** :
    - Lancez le projet Angular et vérifiez que le Web Component fonctionne correctement dans l'application.

Question

- Pourquoi Angular nécessite-t-il la déclaration de `CUSTOM_ELEMENTS_SCHEMA` ? Qu'est-ce que cela signifie pour l'intégration des Web Components ?

Exercice 7 : Utiliser les deux composants précédemment développés (avec open-wc et avec StencilJS) et les exporter dans ce nouveau projet Angular

Objectif

- Intégrer les Web Components créés avec `open-wc` et `StencilJS` dans un projet Angular.
- Exporter ces composants pour qu'ils soient réutilisables dans ce projet Angular.

Instructions

1. **Initialiser un projet Angular** si ce n'est pas déjà fait dans l'exercice précédent.
2. **Importer les Web Components** :
    - Assurez-vous que les fichiers `weather-widget.js` de `open-wc` et `weather-widget.stencil.js` sont correctement importés dans le projet Angular.
    - Testez chaque Web Component dans le projet Angular en les intégrant dans le template de l'application avec un attribut `city`.
3. **Exporter les composants** :
    - Exportez les deux composants en tant que modules JavaScript pour pouvoir les importer dans d'autres projets si nécessaire.
4. **Test du projet** :
    - Lancez l'application et vérifiez que les deux composants fonctionnent correctement dans le projet Angular.

Question

- Quelles difficultés avez-vous rencontrées lors de l'intégration des deux Web Components dans Angular ? Quelles solutions avez-vous trouvées ?

Exercice 8 : Angular Elements

Objectif

- Convertir le Web Component créé dans Angular en un **Angular Element** afin de l'utiliser dans d'autres projets comme un Web Component natif.

- Ce processus permettra de rendre le composant Angular réutilisable en dehors de l'environnement Angular.

Instructions

1. **Créer un composant Angular** similaire à celui créé précédemment dans l'Exercice 6.
2. **Convertir le composant Angular en Angular Element** :
   - Utilisez la bibliothèque `@angular/elements` pour convertir le composant en un Angular Element.
3. **Exporter le Web Component** :
   - Exposez le composant Angular comme un Web Component en l'exportant sous forme de module JavaScript.
4. **Test du projet** :
   - Testez l'intégration du composant Angular Element dans un projet minimaliste HTML pour vérifier qu'il fonctionne comme un Web Component.

Question

- Quels sont les avantages de l'utilisation d'Angular Elements par rapport aux Web Components classiques ?

Exercice 9 : Initier un projet React minimaliste

Objectif

- Créer un projet React minimaliste dans lequel vous allez intégrer le Web Component `weather-widget` créé précédemment.
- Utiliser React pour intégrer un Web Component dans un projet React.

Instructions

1. **Initialiser un projet React minimaliste** en suivant les instructions de la documentation officielle.
   - Commande : `npx create-react-app weather-app`
2. **Importer le Web Component** :
   - Assurez-vous que le fichier `weather-widget.js` est correctement importé dans le projet React.
3. **Utiliser le Web Component dans un composant React** :
   - Utilisez le composant `weather-widget` dans le JSX d'un composant React.
4. **Test du projet** :
   - Lancez le projet React et vérifiez que le Web Component fonctionne correctement dans l'application.

Question

- Quelle est la différence entre l'intégration de Web Components dans un projet React et l'intégration dans un projet Angular ?

Exercice 10 : React

Objectif

- Reproduire l'intégration des deux Web Components créés avec `open-wc` et `StencilJS` dans un projet React.
- Tester que les deux Web Components fonctionnent correctement dans l'application React.

Instructions

1. **Initialiser un projet React minimaliste** si ce n'est pas déjà fait dans l'exercice précédent.
2. **Importer les Web Components** :
   - Assurez-vous que les fichiers `weather-widget.js` de `open-wc` et `weather-widget.stencil.js` sont correctement importés dans le projet React.
3. **Utiliser les Web Components dans React** :
   - Intégrez chaque Web Component dans le JSX du projet React, en passant un attribut `city` pour spécifier la ville.
4. **Test du projet** :
   - Lancez l'application React et vérifiez que les deux Web Components fonctionnent correctement.

Question

- Quels sont les défis que vous avez rencontrés lors de l'intégration de Web Components dans un projet React et comment les avez-vous résolus ?

Exercice 11 : Initier un projet Next.js en suivant la documentation

Objectif

- Créer un projet Next.js minimaliste et intégrer le Web Component `weather-widget` créé précédemment dans le projet Next.js.

Instructions

1. **Initialiser un projet Next.js** en suivant les instructions de la documentation officielle.
   - Commande : `npx create-next-app weather-app`
2. **Importer le Web Component** :

- Assurez-vous que le fichier `weather-widget.js` est correctement importé dans le projet Next.js.

3. **Utiliser le Web Component dans Next.js** :
   - Intégrez le Web Component dans un des composants Next.js, en passant un attribut `city` pour spécifier la ville.
4. **Test du projet** :
   - Lancez le projet Next.js et vérifiez que le Web Component fonctionne correctement dans l'application.

Question

- Quelle est la différence entre l'intégration de Web Components dans un projet Next.js et dans un projet React ?

Exercice 12 : Initier un projet Vue.js en suivant la documentation

Objectif

- Créer un projet Vue.js minimaliste et intégrer le Web Component `weather-widget` créé précédemment dans le projet Vue.js.

Instructions

1. **Initialiser un projet Vue.js** en suivant la documentation officielle.
   - Commande : `npm install vue@next`
   - Créez un projet Vue avec Vue CLI ou Vite.
2. **Importer le Web Component** :
   - Ajoutez le fichier `weather-widget.js` dans le dossier public de votre projet Vue.js.
3. **Utiliser le Web Component dans Vue.js** :
   - Intégrez le Web Component dans le template d'un composant Vue.js, en passant un attribut `city` pour spécifier la ville.
4. **Test du projet** :
   - Lancez le projet Vue.js et vérifiez que le Web Component fonctionne correctement dans l'application.

Question

- Quels sont les principaux avantages et inconvénients de l'intégration des Web Components dans Vue.js par rapport aux autres frameworks ?