Cahier d'Exercices - Introduction à StencilJS

> 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 : Installation et Configuration de StencilJS

Objectifs

- Créer un projet StencilJS.
- Configurer les éléments de base d'un projet Stencil.

Instructions

1. Installez **Node.js** (version LTS recommandée) et **npm** sur votre machine.
2. Créez un nouveau projet StencilJS en utilisant la commande suivante : `bash npm init stencil`
3. Suivez les instructions à l'écran pour initialiser votre projet.
4. Ouvrez le projet dans votre éditeur préféré (recommandé : VS Code).

Questions

Quelles sont les principales répercussions de l'utilisation de StencilJS dans un projet par rapport à un framework classique ?

Exercice 2 : Créer un Web Component simple avec Stencil

Objectifs

- Créer un Web Component de base en utilisant StencilJS.

Instructions

1. Créez un nouveau composant StencilJS `my-greeting` qui affiche un message de bienvenue.
2. Utilisez le décorateur `@Prop` pour ajouter une propriété `name` et afficher "Hello, [name]" dans le composant.
3. Utilisez la commande suivante pour tester le composant :

   ```
   npm start
   ```

Code de base

```
import { Component, Prop } from '@stencil/core';

@Component({
  tag: 'my-greeting',
```

```
  styleUrl: 'my-greeting.css',
  shadow: true,
})
export class MyGreeting {
  @Prop() name: string;

  render() {
    return <p>Hello, {this.name}!</p>;
  }
}
```

Questions

Quelles sont les différences entre `@Prop` et `@State` dans la gestion des données au sein d'un composant ?

Exercice 3 : Ajouter un Service Worker à votre application

Objectifs

- Implémenter un Service Worker dans une application Stencil.

Instructions

1. Ajoutez un fichier `service-worker.js` à votre projet Stencil.
2. Enregistrez ce Service Worker dans votre application via `navigator.serviceWorker.register()`.
3. Assurez-vous que les ressources de base comme `index.html` et `app.js` sont mises en cache et accessibles hors ligne.

Questions

- Expliquez comment un Service Worker peut améliorer la performance d'une application web.

Code de base

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker
    .register('/service-worker.js')
    .then(() => console.log('Service Worker enregistré.'))
    .catch((error) => console.error('Erreur lors de l\'enregistrement :', error));
}
```

Exercice 4 : Validation des formulaires

Objectifs

- Créer un formulaire de saisie avec validation de données.

Instructions

1. Créez un formulaire simple avec des champs `email` et `password`.
2. Implémentez une validation côté client qui empêche l'envoi du formulaire si l'un des champs est vide ou invalide.
3. Ajoutez un message d'erreur qui s'affiche lorsque les validations échouent.

Questions

- Quelle est l'importance de valider les formulaires côté client et côté serveur ?

Code de base

```
validateEmail(email: string): boolean {
  const regex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
  return regex.test(email);
}

handleSubmit(event: Event) {
  event.preventDefault();
  if (!this.validateEmail(this.email)) {
    this.errorMessage = 'Email invalide.';
  } else {
    this.errorMessage = '';
    // Soumettre le formulaire
  }
}
```

Exercice 5 : Gestion du routage avec Stencil

Objectifs

- Configurer le routage dans une application Stencil.

Instructions

1. Installez le module `@stencil/router` dans votre projet Stencil.
2. Définissez deux routes : une pour la page d'accueil (`/`) et une autre pour une page "À propos" (`/about`).
3. Testez le routage en naviguant entre les pages.

Questions

- Quels sont les avantages de l'utilisation d'un routeur dans une application monopage (SPA) ?

Code de base

```
import { Router, Route } from '@stencil/router';

// ...

<Router>
  <Route url="/" component="app-home" />
  <Route url="/about" component="app-about" />
</Router>
```

Exercice 6 : Service Worker avec Workbox

Objectifs

- Utiliser **Workbox** pour simplifier l'implémentation des Service Workers.

Instructions

1. Installez **Workbox** dans votre projet Stencil via npm.
2. Ajoutez un Service Worker qui met en cache les ressources statiques avec la stratégie `CacheFirst`.
3. Testez le fonctionnement du cache en accédant à votre application sans connexion réseau.

Questions

- Comment **Workbox** simplifie-t-il la gestion des Service Workers et du cache ?

Code de base

```
workbox.routing.registerRoute(
  ({ request }) => request.destination === 'image',
  new workbox.strategies.CacheFirst({
    cacheName: 'images-cache',
    plugins: [
      new workbox.expiration.ExpirationPlugin({
        maxEntries: 100,
        maxAgeSeconds: 30 * 24 * 60 * 60, // 30 jours
      }),
    ],
  })
);
```