Web Component : créer des composants Web autonomes et réutilisables

Formation Professionnelle

2024 - v1.0.0

Copyright et Licence

Auteur : Noël Macé
Année de création : 2024 **Contact :** contact@noelmace.com

Votre formateur

Noël Macé

- Développeur / Architecte Senior Fullstack
- Consultant-Formateur expert en développement Web
- **Expérience :**
  - Plus de 15 années dans l'informatique et l'enseignement
  - Grande diversité d'expérience (technologies, contexte et rôle)
  - Focus sur les évolutions des standards et pratiques web
  - nombreuses conférences
  - livre complet publié aux éditions ENI en 2021
- **Motivations :**
  - Partager des connaissances et encourager les bonnes pratiques de développement.
  - Promouvoir les technologies et approches de développement ouvertes, démocratiques et respectueuses des usagers
- **Contact :** contact@noelmace.com

Tour de table

- Prénom
- Poste / métier
- Expérience du sujet abordé
- Attentes

Règles à respecter

1. Répondez à l'appel et/ou **signez la feuille d'émargement** pour chaque demi-journée
2. En cas de retard, attendez la prochaine pause pour rejoindre la salle de cours afin de ne pas perturber la classe
3. Durant une présentation magistrale :
   - 3.1 Levez la main et attendez d'être désigné par le formateur avant d'intervenir / poser une question
   - 3.2 Évitez toute discussion et bruit susceptible de perturber la classe
4. Tout propos ou acte discriminant, violent ou irrespectueux envers toute personne impliquée dans la formation amènera à une exclusion immédiate et automatique du cours et sera reportée à l'organisme de formation

Objectifs de cette formation

- Vous initier à l'univers des Web Components.
- Explorer leurs applications dans vos projets.
- Développer des composants robustes et interopérables.

Contexte et déroulement

- **Audience :** Développeurs ayant une connaissance de base en HTML, CSS, JavaScript.
- **Structure :**
  1. La norme des Web Components.
  2. Rappels DOM & AJAX.
  3. Template HTML à chargement différé.
  4. Shadow DOM et CSS, les fragments de documents.
  5. CSS : le besoin d'encapsulation.
  6. Custom Elements.
- **Déroulement :**
  1. Présentations théoriques.
  2. Démonstrations.
  3. Discussions et Q/A.
  4. Exercices pratiques.
  5. Discussions et Q/A.
  6. GOTO 1

La norme des Web Components

Qu'est-ce qu'un Web Component ?

- Ensemble de standards du W3C pour créer des éléments HTML personnalisés.
- Réutilisables et autonomes.
- Indépendants des frameworks JavaScript.

Standards des Web Components

1. **Custom Elements**
   Permet de définir des balises HTML personnalisées.
2. **Shadow DOM**
   Fournit un encapsulage des styles et du DOM.
3. **HTML Templates**
   Permet de créer des fragments HTML réutilisables.
4. ***HTML Imports*** *(obsolète)*
   Charge des ressources HTML externes (non recommandé).

Rappels DOM & AJAX

Document Object Model (DOM)

- **DOM :** Représentation arborescente des documents HTML.
- Permet d'accéder et de manipuler les éléments d'une page via JavaScript.
- Exemples :
  - `document.querySelector()`
  - `element.setAttribute()`

AJAX : Asynchronous JavaScript And XML

- **AJAX :** Technique pour charger des données dynamiquement sans recharger la page.
- Utilise des API comme :
  - `XMLHttpRequest`
  - `fetch()`
- Exemples d'applications :
  - Chargement de contenu.
  - Envoi de données à un serveur.

Template HTML à chargement différé

Le rôle des templates HTML

- Contiennent du contenu HTML réutilisable.
- Utilisent la balise `<template>` :

```
<template id="my-template">
  <p>Bonjour, je suis un template !</p>
</template>
```

Template HTML à chargement différé

Le rôle des templates HTML

- Contiennent du contenu HTML réutilisable.
- Utilisent la balise `<template>` :

```
<template id="my-template">
  <p>Bonjour, je suis un template !</p>
</template>
```

Utilisation des templates

1. Sélectionner le template :

```
const template = document.getElementById('my-template')
```

2. Cloner son contenu :

```
const clone = template.content.cloneNode(true);
```

3. Ajouter au DOM :

```
document.body.appendChild(clone);
```

Shadow DOM et CSS : Encapsulation

Le Shadow DOM, c'est quoi ?

- Propose un DOM encapsulé pour les composants.
- Les styles et le DOM interne ne fuient pas vers l'extérieur.
- Exemple d'utilisation :

```
const shadow = element.attachShadow({ mode: 'open' });
shadow.innerHTML = `<style>p { color: red; }</style><p>
```

Avantages du Shadow DOM

- Encapsulation : Styles internes non affectés par les styles globaux.
- Isolation : Préserve l'intégrité des composants.
- Exemples pratiques :
  - Composants réutilisables.
  - Widgets indépendants.

CSS : Le besoin d'encapsulation

Problème des styles globaux

- Les styles CSS affectent souvent tout le document.
- Risques :
  - **Conflits de styles** entre composants.
  - **Effets de bord** imprévus sur d'autres parties de la page.

Solution : Shadow DOM + Scoped Styles

- Le Shadow DOM encapsule les styles et isole les composants.
- Exemple pratique :

```
class StyledButton extends HTMLElement {
  constructor() {
    super();
    const shadow = this.attachShadow({ mode: 'open' });
    shadow.innerHTML = `
                <style>
                    .btn {
                        color: white;
                        background-color: blue;
                        border: none;
                        padding: 10px;
                        cursor: pointer;
                    }
                    .btn:hover {
                        background-color: darkblue;
                    }
                </style>
```

Résultat attendu

- Encapsulation complète : Les styles du bouton ne sont pas affectés par ceux de la page principale.
- Portabilité : Les composants peuvent être utilisés sur différentes pages ou projets sans ajustements.

Custom Elements

Qu'est-ce qu'un Custom Element ?

- Permet de créer de nouvelles balises HTML personnalisées.
- Étend la classe `HTMLElement` avec un comportement et des styles propres.

Syntaxe de base

1. Définir une classe pour l'élément :

```
class MyElement extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });
    this.shadowRoot.innerHTML = `<p>Bonjour, Web Compor
  }
}
```

2. Enregistrer l'élément :

```
customElements.define('my-element', MyElement);
```

Exemple d'utilisation

Une fois défini, l'élément peut être utilisé dans une page HTML :

```
<my-element></my-element>
```

Méthodes du cycle de vie

- `connectedCallback` : Appelé lorsque l'élément est inséré dans le DOM.
- `disconnectedCallback` : Appelé lorsque l'élément est retiré du DOM.
- `attributeChangedCallback` : Appelé lorsqu'un attribut de l'élément change.

Exemple : Cycle de vie en action

```
class LifecycleElement extends HTMLElement {
  connectedCallback() {
    console.log('Element inséré dans le DOM');
  }

  disconnectedCallback() {
    console.log('Element retiré du DOM');
  }

  static get observedAttributes() {
    return ['data-example'];
  }

  attributeChangedCallback(name, oldValue, newValue) {
    console.log(`Attribut ${name} changé de ${oldValue} à $
  }
}

// Enregistrement du Custom Element
```

Résultat attendu

1. Lors de l'insertion de l'élément dans le DOM : `Element inséré dans le DOM`
2. Lors du changement d'attribut : `Attribut data-example changé de null à test-value`
3. Lors du retrait de l'élément du DOM : `Element retiré du DOM`

Avantages des Web Components

Pourquoi utiliser les Web Components ?

1. **Réutilisabilité :**
   - Les composants peuvent être utilisés sur plusieurs projets sans modification.
   - Exemples : boutons, cartes, modales.
2. **Encapsulation :**
   - Protection des styles et du comportement.
   - Aucune interférence avec les styles globaux.
3. **Interopérabilité :**
   - Fonctionnent avec tous les frameworks JavaScript (React, Angular, Vue, etc.).
   - Compatibles avec du JavaScript natif.
4. **Maintenance facilitée :**
   - Code plus modulaire.
   - Séparation claire des responsabilités (HTML, CSS, JS).

Limites actuelles

1. **Performance :**
   - Le Shadow DOM peut ajouter une légère surcharge si utilisé massivement.
2. **Compatibilité :**
   - Nécessite des polyfills pour certains anciens navigateurs.
3. **Complexité initiale :**
   - Courbe d'apprentissage plus élevée pour les débutants.

Étape pratique : Créer un composant simple

Objectif

Créer un composant personnalisable pour afficher un message d'alerte.

Étape 1 : Définir le composant

1. **Code HTML :**

```
<alert-box type="success"> Opération réussie ! </alert-box>
```

2. **Code JavaScript :**

```
class AlertBox extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });
  }

  connectedCallback() {
    const type = this.getAttribute('type') || 'info';
    const message = this.innerHTML;

    this.shadowRoot.innerHTML = `
            <style>
                .alert {
                    padding: 10px;
                    margin: 5px 0;
                    border-radius: 4px;
```

Étape 2 : Tester dans le navigateur

1. Ajoutez le script dans une page HTML.
2. Insérez une balise `<alert-box>` avec différents types (success, error, info).
3. Observez le rendu et testez la personnalisation.

Web Components et frameworks JavaScript

Intégration avec des frameworks modernes

1. **React :**
   - Utilisez un Custom Element comme une balise standard.
   - Exemple :

```
function App() {
  return <my-element />;
}
```

2. **Vue.js :**
   - Utilisez directement les Web Components dans les templates Vue.
   - Exemple :

```
<template>
  <my-element></my-element>
</template>
```

3. **Angular :**
   - Activez `schemas: [CUSTOM_ELEMENTS_SCHEMA]` dans le module Angular.
   - Exemple :

```
@NgModule({
    declarations: [...],
```

Bénéfices

- Les Web Components fonctionnent indépendamment du framework.
- Faciles à intégrer dans des applications existantes.

Avantages et limites des Web Components

Points forts

1. **Réutilisabilité :** Partager des composants entre projets.
2. **Interopérabilité :** Fonctionnent avec ou sans framework.
3. **Encapsulation :** Protection des styles et comportement.

Limites

1. **Support des anciens navigateurs :** Polyfills nécessaires.
2. **Complexité initiale :** Apprentissage des nouvelles API.
3. **Performance :** Shadow DOM peut entraîner une légère surcharge.

Perspectives

- **Adoption croissante :** Les Web Components gagnent en popularité grâce à leur simplicité et leur flexibilité.
- **Écosystème :** Intégration dans des outils modernes comme Storybook ou des librairies spécialisées.
- **Standards en évolution :** Les API continueront de s'améliorer avec le temps.

Rôle des développeurs

- Encourager l'adoption des Web Components dans vos projets.
- Contribuer à l'écosystème en développant des composants open source.
- Continuer d'explorer les nouvelles fonctionnalités des standards W3C.

Rappel des sujets abordés

1. La norme des Web Components
   - Les standards : **Custom Elements**, **Shadow DOM**, **HTML Templates**.
   - Objectif : Créer des composants autonomes, réutilisables et encapsulés.

2. Concepts fondamentaux

- **DOM & AJAX :** Comprendre le fonctionnement du DOM et la manipulation asynchrone des données.
- **Templates HTML :** Créer et utiliser des fragments HTML réutilisables.
- **Shadow DOM :** Encapsuler le DOM et les styles pour éviter les conflits.
- **Custom Elements :** Étendre les fonctionnalités d'HTML avec des balises personnalisées.

3. Pratique et cas d'usage

- Exemple d'un composant d'alerte avec styles encapsulés.
- Utilisation des Web Components avec des frameworks modernes comme React, Vue et Angular.
- Gestion du cycle de vie des Custom Elements.

4. Mise en perspective

- avantages et inconvénients
- perspectives
- rôle des développeurs

Questions ?

- Des points nécessitent-ils plus d'explications ?
- Avez-vous des cas pratiques ou des projets où intégrer les Web Components ?
- Cette formation a-t-elle répondu à vos attentes ?
- Avez-vous des retours sur cette formation ?

Merci pour votre attention

N'OUBLIEZ PAS !!!

- D'**ENREGISTRER** vos travaux
- De bien avoir **SIGNÉ LA FEUILLE D'ÉMARGEMENT** avant de partir
- *(si applicable)* De **COMPLÉTER LA FICHE D'ÉVALUATION** qui vous a été remise
- De récapituler le cours dans votre tête ou parcourir vos notes avant d'aller dormir ce soir.