

Composant web (Web Component)

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`

Composant web ?

- Ensemble de plusieurs technologies Web : **HTML**, **CSS**, **JavaScript** (**TypeScript**)...
- Permettant de définir des balises personnalisées et réutilisables
- En cours de normalisation par le **W3C**

En parlant composant Web

- Custom Element
- Shadow Root
- Scoped CSS

Démarche

- Créer un répertoire `cours-composant-web` dans votre espace de travail
- Lancer **VSC** et allez dans `File > Open Folder...` et choisir `cours-composant-web`
- Dans `cours-composant-web`, créer une page `index.html` contenant la nouvelle balise (ainsi que le script `ts` ou `js`)

Composant Web

Exemple : créons la page HTML suivante :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
  </body>
</html>
```

Composant Web

Exemple : créons la page HTML suivante :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
  </body>
</html>
```

Objectif

Introduire notre propre balise

Composant Web

Ajoutons notre nouvelle balise

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
  </body>
</html>
```

Composant Web

Ajoutons notre nouvelle balise

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
  </body>
</html>
```

Remarque

Le nom de la nouvelle balise doit contenir un tiret '-'.

Préparons la classe associée à cette balise

Notre classe doit

- hériter de la classe `HTMLElement`
- définir les trois méthodes suivantes qui représentent le cycle de vie de notre balise
 - `connectedCallback`,
 - `disConnectedCallback` **et**
 - `attributeChangedCallback`.

Ensuite il faut associer la classe à la balise

Composant Web

Créons notre classe associée (dans un fichier `ma-balise.ts`)

```
class MaBalise extends HTMLElement {
  constructor() {
    console.log("constructor");
    super();
  }
  connectedCallback() {
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise', MaBalise);
```

Mettons à jour le fichier `index.html` (intégration du script)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
    <script src="ma-balise.ts"></script>
  </body>
</html>
```

On peut modifier le contenu et utiliser les fonctions JavaScript sur le DOM

```
class MaBalise extends HTMLElement{
  constructor() {
    super();
    this.innerHTML += ` from TS`;
  }
  connectedCallback() {
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise', MaBalise)
```

On peut même définir le contenu de notre élément dans une balise `<p>`

```
class MaBalise extends HTMLElement{
  constructor() {
    super();
    this.innerHTML = `
    <style>p { background-color: red; color: white; }</
    style>
    <p>${ this.innerHTML }</p>`;
  }
  connectedCallback() {
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise', MaBalise)
```

Composant Web

Un petit problème : si on ajoute un deuxième paragraphe après le composant Web, il sera aussi affiché en rouge

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
    <p> bonsoir </p>
    <script src="ma-balise.ts"></script>
  </body>
</html>
```

Pour résoudre ce problème : Shadow DOM

- Dans notre composant Web, on crée une racine ombre (`shadow-root`) et on lui attache les éléments
- Pour cela, on utilise une méthode `attachShadow (mode : value) : value` peut être
 - `open` : on peut le cibler avec JavaScript
 - `closed` : on ne peut le cibler avec JavaScript

Définissons maintenant le Shadow DOM

```
class MaBalise extends HTMLElement{
  constructor(){
    super();
    let shadow = this.attachShadow({ mode: 'open' });
    shadow.innerHTML = `
    <style>p { background-color: red; color: white; }</style>
    <p>${ this.innerHTML }</p>`;
  }
  connectedCallback(){
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise',MaBalise)
```

Le deuxième paragraphe n'est plus affiché en rouge

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
    <p> bonsoir </p>
    <script src="ma-balise.ts"></script>
  </body>
</html>
```

Le deuxième paragraphe n'est plus affiché en rouge

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
    <p> bonsoir </p>
    <script src="ma-balise.ts"></script>
  </body>
</html>
```

Aller vérifier dans le DOM qu'un nœud `shadow-root` a été ajouté.

Composant Web

Définissons une classe CSS bleu

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Composant web</title>
    <style>.bleu { color: skyblue; }</style>
  </head>
  <body>
    <h1>Ici, on introduit une nouvelle balise</h1>
    <ma-balise>Bonjour</ma-balise>
    <p class=bleu> bonsoir </p>
    <script src="ma-balise.ts"></script>
  </body>
</html>
```

Essayons d'utiliser cette classe bleu dans notre composant Web

```
class MaBalise extends HTMLElement{
  constructor() {
    super();
    let shadow = this.attachShadow({ mode: 'open' });
    shadow.innerHTML = `
    <p class=bleu>${ this.innerHTML }</p>`;
  }
  connectedCallback() {
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise', MaBalise);
```

Essayons d'utiliser cette classe bleu dans notre composant Web

```
class MaBalise extends HTMLElement{
  constructor() {
    super();
    let shadow = this.attachShadow({ mode: 'open' });
    shadow.innerHTML = `
    <p class=bleu>${ this.innerHTML }</p>`;
  }
  connectedCallback() {
    console.log("connected");
  }
  disconnectedCallback() {
    console.log("disconnected");
  }
  attributeChangedCallback() {
    console.log("ACD");
  }
}
window.customElements.define('ma-balise', MaBalise);
```

Aller vérifier mais le paragraphe Bonjour sera affiché en noir

Remarque

- Dans un ShadowRoot :
 - on ne propage pas le **CSS** aux autres éléments (définis après le composant Web)
 - on ne récupère pas le **CSS** défini pour les autres éléments
- ⇒ **Scoped CSS**

Pour accéder à notre paragraphe depuis un autre fichier JavaScript

```
let val = document.querySelector('ma-balise').  
    shadowRoot.querySelector('p').innerHTML;  
console.log(val);  
// affiche Bonjour
```

© Achref L.

Pour accéder à notre paragraphe depuis un autre fichier JavaScript

```
let val = document.querySelector('ma-balise').  
    shadowRoot.querySelector('p').innerHTML;  
console.log(val);  
// affiche Bonjour
```

En mettant `mode à closed` dans `ma-balise.ts`, le code précédent génère une erreur

```
let shadow = this.attachShadow({ mode: 'closed' });
```