

Développement front

MMI 3 – TP#3 S5

Danielo **JEAN-LOUIS**

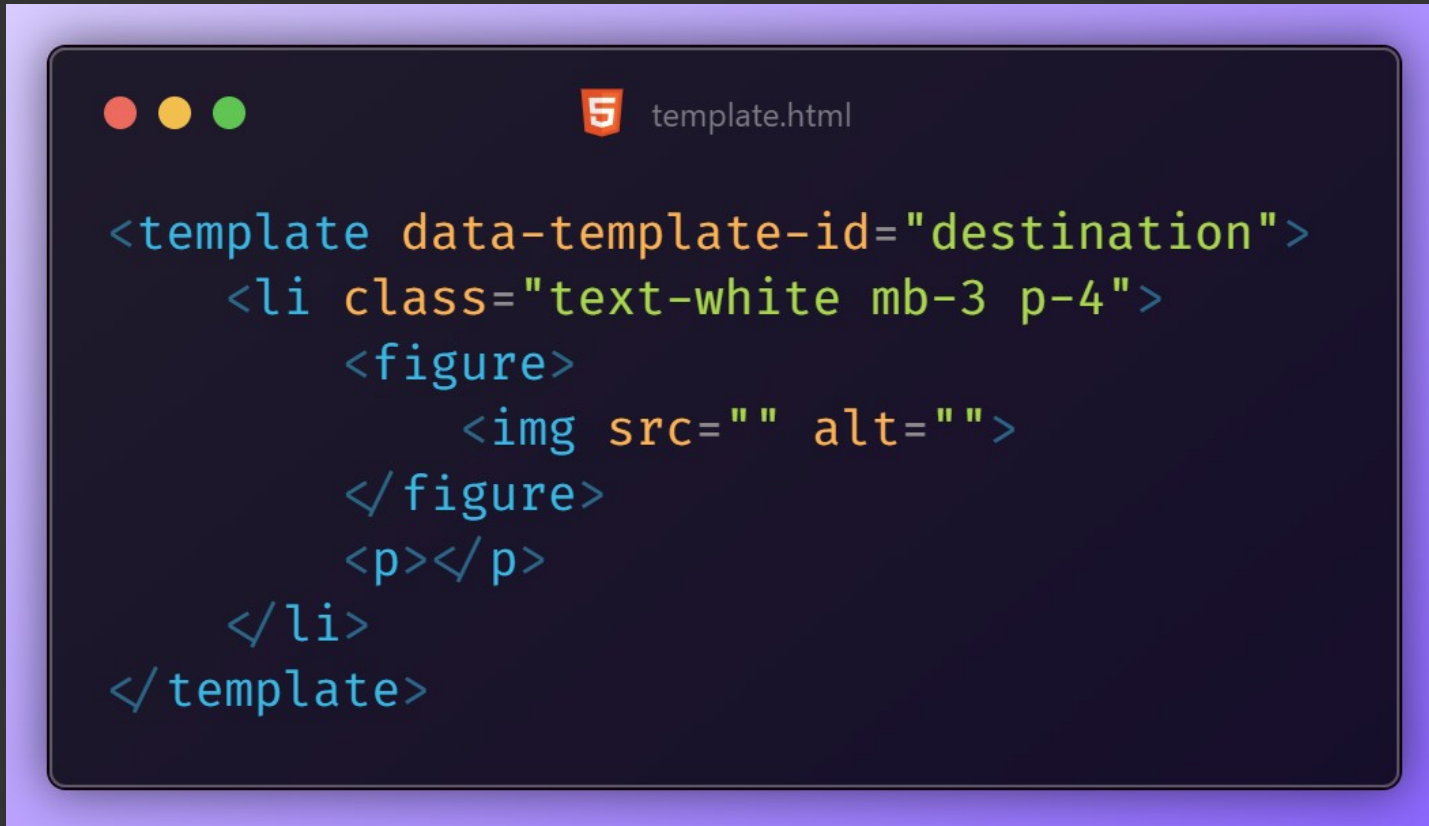
Balise <template>

- Nouveauté HTML5 gérée par tous les navigateurs
- Permet de définir des gabarits de code HTML
 - Affichés et remplis ensuite en javascript
- **Contenu non affiché/chargé par le navigateur**
 - Aucune incidence sur le temps de chargement

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Balise <template>

A code editor window with a dark blue background and a light blue border. The title bar shows three colored circles (red, yellow, green) and a red icon with a white 'S' followed by the text 'template.html'. The code is written in a light blue monospace font and is indented to show a nested structure.

```
<template data-template-id="destination">
  <li class="text-white mb-3 p-4">
    <figure>
      <img src="" alt="">
    </figure>
    <p></p>
  </li>
</template>
```

Définition d'un template "squelette" à destination d'une balise /

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Balise <template>

- Peut avoir du contenu prédéfini :
 - Lien, CSS, script, texte...
- Doit respecter les règles du code HTML vues jusqu'à présent
- Pas de limite d'imbrications ou de template sur une page

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Balise <template>

- Limite les erreurs lors d'ajouts de nouvelles balises dans une page HTML
- Réutilisable
- Solution de substitution à la place d'un framework javascript

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Balise <template>

```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <template id="tpl">
      #document-fragment
    </template>
    <script>
    </script>
    <p class="message">Bonjour !</p>
  </body>
</html>
```

Le template est grisé dans la console du navigateur, il n'est donc pas affiché dans la page

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Balise <template>

```
● ● ●  
  
<template data-tpl-id="tpl">  
  <p class="message">Bonjour !</p>  
</template>  
  
<script>  
  const tpl = document.querySelector("[data-tpl-id=tpl]")  
  // Ajout du contenu du template dans la page  
  // on clone le template pour pouvoir le réutiliser  
  document.body.append(tpl.content.cloneNode(true));  
</script>
```

Ce code permet d'afficher un template contenant une balise <p> avec du texte.

content.cloneNode() permet de cloner le contenu du template

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>
- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Méthode .append()

- Méthode javascript
- Permet d'ajouter une balise ou du texte à l'intérieur d'une autre balise
- Utilisable sur n'importe quelle balise HTML
 - A condition qu'elle puisse accepter du contenu

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Element/append>

Méthode .cloneNode()

- Méthode javascript
- Copie le nœud sur lequel la méthode a été appelée
- Utilisable sur n'importe quelle balise
- document.importNode est une alternative
- **Ne copie pas les évènements**

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>
- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Méthode .cloneNode()

```
const element = document.querySelector("selector");  
const elementCopy = element.cloneNode(true);
```

On clone une balise pour la réutiliser plus tard

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Balise <template>

```
const newDiv = document.createElement("div");
const newImage = document.createElement("img");
newImage.setAttribute("src", "[...]");
newImage.setAttribute("alt", "");

newDiv.appendChild(newImage);
```

Les deux codes font la même chose, mais le javascript pur est plus verbeux

```
<template>
  <div>
    <img src="" alt="">
  </div>
</template>
```

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/template>

Pratiquons ! - Template (Partie 1)

Pré-requis :

- Avoir la ressource ressources/template

A télécharger ici :

https://github.com/DanYellow/cours/raw/refs/heads/main/developement-front-s5/travaux-pratiques/numero-3/developement-front-s5_travaux-pratiques_numero-3.ressources.zip

Bonnes pratiques

- Nommez bien vos templates
 - Utilisation d'id ou de data-attribute pour cibler vos éléments

Récupération d'éléments

- `.querySelector()`
 - Retourne le premier élément trouvé
- `.querySelectorAll()`
 - Retourne **tous** les éléments trouvés
- Utilise la syntaxe des sélecteurs CSS

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector>
- <https://developer.mozilla.org/fr/docs/Web/API/Document/querySelectorAll>

Récupération d'éléments

- Utilisable sur n'importe quelle balise HTML
 - Limite la portée de la méthode
- Évitez les méthodes : `getElementById()` ou `getElementsByTagName()`, elles sont moins polyvalentes et modernes

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector>
- <https://developer.mozilla.org/fr/docs/Web/API/Document/querySelectorAll>

Récupération d'éléments

```
● ● ●  
  
// [...]  
const item = tpl.content.cloneNode(true);  
const paragraph = item.querySelector("p");  
paragraph.textContent = "mon texte";  
  
document.body.append(item);
```

Ici, nous récupérons et modifions le texte de la balise <p> contenue dans un template
Note : la modification doit se faire **avant** la méthode append()

Pratiquons ! - Template (Partie 2)

Pré-requis :

- Avoir la ressource ressources/template

A télécharger ici :

https://github.com/DanYellow/cours/raw/refs/heads/main/developement-front-s5/travaux-pratiques/numero-3/developement-front-s5_travaux-pratiques_numero-3.ressources.zip

Méthode `.cloneNode()` - Suite

- Accepte un booléen en paramètre
 - `true` : clone profond, tous les enfants sont clonés
 - `false` : clone simple, ne clone que la première balise du template
- **Ne clone pas les évènements js**

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Méthode `.cloneNode()` - Suite

- Évitez de mettre des id dans vos templates car ils seront également dupliqués
 - Et on ne peut (toujours) pas avoir des ids identiques

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Méthode .cloneNode() - Suite

```
● ● ●  
  
<template data-tpl-id="tpl">  
  <article class="mt-3">  
    <p class="bg-red-200"></p>  
  </article>  
</template>  
  
<script>  
  const tpl = document.querySelector("[data-tpl-id=tpl]");  
  const tplCopy = tpl.content.cloneNode(true)  
</script>
```

La variable `tplCopy` contient un clone profond du template.

Ainsi, la balise `<p>` est dans la variable.

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Méthode .cloneNode() - Suite

La variable tplCopy est un clone non-profond du template

Ainsi, la balise <p> **n'est pas** dans la variable et donc inaccessible via `querySelector()`.

```
<template data-tpl-id="tpl">
  <article class="mt-3">
    <p class="bg-red-200"></p>
  </article>
</template>

<script>
  const tpl = document.querySelector("[data-tpl-id=tpl]");
  const tplCopy = tpl.content.cloneNode(false)
</script>
```

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/API/Node/cloneNode>

Pratiquons ! - Template (Partie 3)

Pré-requis :

- Avoir la ressource ressources/template

A télécharger ici :

https://github.com/DanYellow/cours/raw/refs/heads/main/developement-front-s5/travaux-pratiques/numero-3/developement-front-s5_travaux-pratiques_numero-3.ressources.zip

Questions ?

