Générez le contenu de votre page grâce au DOM

7–9 minutes

Les données des pièces automobiles au format JSON sont prêtes à être utilisées. Nous pouvons commencer le développement de notre page web pour Les Bonnes Pièces. 🚀

Notre objectif ici est de générer la fiche d'un produit : l'ampoule LED. Cela se passera en deux étapes :

1. Générer les éléments du DOM.
2. Vérifier la validité des données.

Importez les données de notre fichier JSON

Pour manipuler les données du document JSON que nous avons créé, nous devons d'abord importer ces données dans notre code. Pour y parvenir, nous allons utiliser la fonction fetch de JavaScript dans notre fichier piece.js.

Nous reviendrons plus en détail sur le fonctionnement de la fonction fetch dans la troisième partie de ce cours. Par souci de simplicité, nous nous contenterons pour le moment de rédiger les lignes de code ci-dessous :

```
// Récupération des pièces depuis le fichier JSON
const reponse = await fetch('pieces-autos.json');
const pieces = await reponse.json();
```

Créez de nouveaux éléments du DOM

Dans ce chapitre, nous allons interagir avec le **DOM**. En tant que développeur JavaScript, j'imagine que vous en avez déjà entendu parler 😉. Pour rappel, il s'agit d'une structure en mémoire gérée par le navigateur. Elle a pour but de faciliter la manipulation des éléments graphiques affichés à l'écran. Cela sert essentiellement à transformer une page web statique en une page web "active" capable de réagir aux actions de l'utilisateur et de se mettre à jour.

Créez des éléments

Dans un premier temps, nous allons donc créer la fiche produit d'une ampoule LED. Cela se déroulera en plusieurs étapes :

- choisir les balises HTML adaptées aux informations à afficher ;
- créer les éléments à proprement parler ;
- afficher ces éléments sur la page.

Commençons par choisir nos balises ! Le produit est composé d'un nom, d'un prix, d'une catégorie et d'une image. Les balises HTML les plus adaptées pour chacunes de ces informations sont :

- `img` pour l'image ;
- `h2` pour le nom (on considère que h1 servira au titre du document) ;
- `p` pour le prix et la catégorie.

Créons donc ces quatre balises avec createElement :

```
const article = pieces[0];

const imageElement = document.createElement("img");

imageElement.src = article.image;

const nomElement = document.createElement("h2");

nomElement.innerText = article.nom;

const prixElement = document.createElement("p");

prixElement.innerText = `Prix: ${article.prix} €`;
```

```
const categorieElement = document.createElement("p");

categorieElement.innerText = article.categorie;
```

Heuu…. La syntaxe à la ligne 7, elle est pas un peu bizarre ?

Effectivement, c'est une syntaxe un peu particulière. 😃 C'est ce qu'on appelle des **littéraux de gabarit** (ou template strings, en anglais). Ces derniers permettent de concaténer plus facilement des chaînes de caractères et des variables. Ils commencent et se terminent toujours par des backticks ` et sont composés de variables ou expressions JavaScript sous la forme ${monExpressionJS}.

Par exemple, ` l'addition est de ${2 +3} euros` affichera : "l'addition est de 5 euros".

Dans notre code, cela équivaudra à une concaténation de chaîne classique :

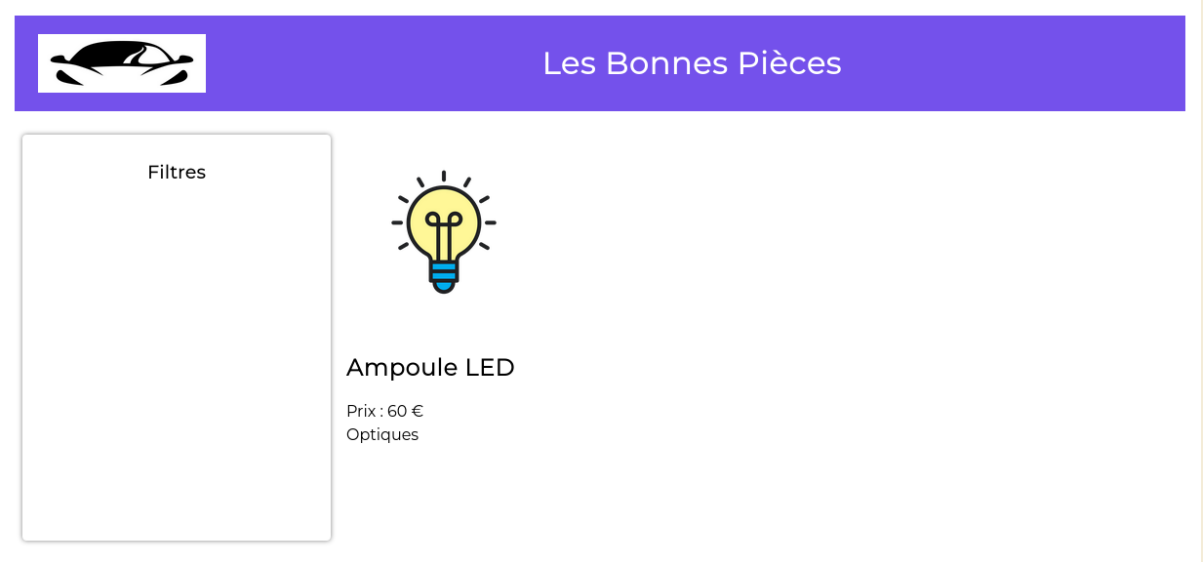
```
prixElement.innerText = "Prix: " + article.prix + "
€";
```

Pour faire ce rattachement, nous avons besoin d'un **parent**. En effet, le DOM structure les éléments sous forme d'arbre avec des enfants et des parents. Il faut donc trouver un parent pour accueillir nos nouveaux éléments. Ainsi, nous allons utiliser la fonction appendChild en JavaScript.

Notre page web contient une balise section avec la classe "fiches" que nous utiliserons comme parent. Nous la récupérons grâce à querySelector :

```
const sectionFiches =
document.querySelector(".fiches");

sectionFiches.appendChild(imageElement);

sectionFiches.appendChild(nomElement);

sectionFiches.appendChild(prixElement);

sectionFiches.appendChild(categorieElement);
```

Et voici le résultat !



La fiche produit “Ampoule LED” a été ajoutée au site des Bonnes Pièces.

Vérifiez les données

Vous avez créé votre première fiche produit. Bravo ! 🥳 Mais vous n’en avez pas encore tout à fait fini ! Il vous reste deux problématiques à résoudre :

- votre client souhaite afficher un indicateur de prix pour aider les consommateurs à faire leur choix ;
- lorsqu'on affiche la pièce “Balai d'essuie-glace", la catégorie n’est pas renseignée.

Regardons ensemble comment adapter l’affichage de cette fiche en fonction de ces deux situations !

Choisir entre deux possibilités grâce à l’opérateur ternaire

L'**opérateur ternaire** s'utilise lorsque l'on doit choisir entre deux possibilités. Dans notre cas, nous voulons afficher un indicateur à côté du prix pour indiquer au client si l'article est abordable ou non. Si un article vaut moins de 35 euros, alors on considère qu'il est abordable, et on affichera un seul symbole euro. À l'inverse, si l'on considère que le prix est élevé, on affichera trois symboles euro.

Dans notre cas, on écrira donc :

```
article.prix < 35 ? "€" : "€€€"
```

Reprenons le code précédent et modifions la ligne de code qui ajoute le paragraphe du prix :

```
// …

const prixElement = document.createElement("p");

prixElement.innerText = `Prix: ${article.prix} €
(${article.prix < 35 ? "€" : "€€€"})`;

// ...

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

Fournir une valeur par défaut grâce à l'opérateur nullish

L'opérateur nullish s'utilise lorsque vous pensez avoir une information dans une variable mais que finalement, il n'y en a pas.

Ça peut arriver quand, concrètement ? 🤨

Eh bien, quand une valeur est null, et donc qu'elle n'a pas de valeur, ou bien lorsqu'elle est undefined, et donc qu'elle n'est pas définie. Dans notre cas, la pièce automobile "Balai d'essuie-glace" n'appartient à aucune catégorie. On aimerait le préciser entre parenthèses lorsque c'est le cas.

On écrira donc :

```
categorieElement.innerText = article.categorie ??
"(aucune catégorie)";
```

Récapitulons en vidéo

Vous pouvez revoir les différentes étapes de ce chapitre dans la vidéo ci-dessous :

À vous de jouer

À partir de la branche GitHub P1C4-Exercice :

Éditez le fichier pieces.js, ajoutez-y le code développé dans ce chapitre de manière à générer une fiche produit, et ajoutez :

- un élément de paragraphe pour la description avec le texte “Pas de description pour le moment.” en cas d’absence de description ;
- un élément de paragraphe avec le texte “En stock” si l’article est disponible, ou “Rupture de stock” s’il n’est plus disponible.

Corrigé

Vous pouvez vérifier votre travail en consultant la branche GitHub P1C4-Solution et en regardant la vidéo ci-dessous :

En résumé

- En manipulant le DOM, vous pouvez :
- créer de nouveaux éléments avec createElement ;
- afficher les nouveaux éléments avec appendChild.
- Utilisez des opérateurs pour vérifier vos données avant de les afficher :
- l’opérateur ternaire pour transformer une valeur en une autre selon 2 possibilités ;
- l’opérateur nullish pour fournir une valeur de remplacement quand une valeur est null, ou bien lorsqu’elle est undefined.

Et voilà ! Vous venez d’apprendre à générer une page web à partir de données existantes, à l’aide du JSON et du DOM. Toutes mes félicitations ! 🥳 Attachez vos ceintures, la prochaine étape dans la

deuxième partie de ce cours est de rendre cette page interactive ! Mais avant, je vous propose de tester vos connaissances grâce au quiz.