> **"Je vous écris une longue lettre parce que je n'ai pas le temps d'en écrire une courte."**

Blaise Pascal

Prérequis : lecture

🪛Nous allons interroger un arbre généalogique.

Présence d'une BD[1] :

L'arbre généalogique est stocké dans une base de données. L'accès aux objets de la BD sera remplacé par l'accès à un fichier JSON[2].

Réalisation d'un prototype

Imaginez que vous soyez chef de projet, vous rédigez en urgence un prototype pour les membres de votre équipe. Ce prototype permet à tous de fixer les idées.

👷 Commençons par réaliser un prototype en JS permettant d'afficher les personnes nés après une date : Affichage des personnes nés après 1900)[3]

→codepen

Analyse du code

HTML

Le HTML est quasi vide, <section> sert à afficher des résultats.

HTML
<body> <section></section> </body>

CSS

Je remarque que l'équipe qui gère le CSS maîtrise la technologie flex.

---

[1] Mais, oublions la gestion de la BD qui n'est pas l'objet de ce cours.
[2] Il me restera à étudier l'objet JSON et ses deux méthodes

JSON.parse()

JSON.stringify()

[3] avec destructuration

CSS

```
article {
   flex : 0 0 25%;
   max-width: 25%;
   padding: 5px;
   display : flex;
   flex-direction : column;
   align-items : center;
   justify-content : center;
   border-top : 1px solid;
}
```

Notez la modification possible pour obtenir 6 articles par ligne.

```
article {
   flex : 0 0 25%;
   max-width: 25%;
   padding: 5px;
   display : flex;
   flex-direction : column;
   align-items : center;
   justify-content : center;
   border-top : 1px solid;
}
```

```
article {
   flex : 0 0 16.66%;
   max-width: 16.66%;
   padding: 5px;
   display : flex;
   flex-direction : column;
   align-items : center;
   justify-content : center;
   border-top : 1px solid;
}
```

JS

Coté JS, je vois l'utilisation d'API[4] (fetch, classList et DOM)

Nous étudierons certainement l'API fetch (mais il nous faudra découvrir avant ce qu'est une promesse !). Mais pour l'instant, ce qui semble essentiel est en lig. 7.

Lig.7, nous disposons d'un tableau (people) qui est l'ensemble des personnes contenu dans la base. Ce tableau est passé en paramètre à une fonction pour manipulation.

[4] Les APIs (Application Programming Interfaces) sont des constructions disponibles dans les langages de programmation pour permettre aux développeurs de créer plus facilement des fonctionnalités complexes. Elles s'occupent des parties de code plus complexes, fournissant au développeur une syntaxe plus facile à utiliser à la place.MDN

```
1.  fetch('https://output.jsbin.com/fokuyi/1.js')
2.      .then(function (response) {
3.          return response.json()
4.      })
5.      .then(function (json) {
6.          let people = json;
7.          initialize(people);
8.      }).catch(function (err) {
9.          console.log('Fetch problem: ' + err.message);
10.     });
11.
12. function initialize(people) {
13.
14.     let sectionUI = document.body.querySelector("section");
15.
16.     for (let i = 0; i < people.length - 1; i++) {
17.
18.         let articleUI =
19.             `<article>
20.                 <div class="">${people[i].name}</div>
21.                 <div class="">${i}</div>
22.             </article>`;
23.
24.         sectionUI.insertAdjacentHTML("beforeEnd", articleUI);
25.
26.
27.         if (people[i].born > "1900") {
28.             console.log("find")
29.             sectionUI.lastChild.classList.add("selected");
30.         }
31.     }
32. }
```

Finalement si l'on oublie les APIs, le code se réduit à la définition d'une fonction lig. 1 et à une boucle sur un tableau *people lig.2*.

```
1.  function initialize(people){
2.      for (let i = 0; i < people.length - 1; i++) {
```

```
3.          if (people[i].born > "1900") {}
4.      }
5.  }
```

Nous pouvons utiliser avantageusement la destructuration :

Affichage des personnes nées après 1830

Objectif du projet :

Si l'objectif en lui-même n'est pas clair, l'idée générale est de pouvoir afficher une extraction de la BD en fonction de contraintes.

Par exemple, comment transformer mon code JS, si je désire afficher les hommes et les femmes nés après 1900, ou encore les femmes qui sont mortes avant 45 ans …

On peut s'amuser à une rapide comparaison du code JS

Affichage des personnes nés après 1900	Affichage des hommes

for (let i=0; i<people.length-1; i++){
 	
if (people[i].born > "1900") {
Affichage}
}
	for (let i=0; i<people.length-1; i++){
	
	if (people[i].sex === "m") {
	Affichage}
	}

Nous serons jugé sur la qualité de notre code.

Si je ne me préoccupe pas pour l'instant de l'accès à la BD, ni de l'interface graphique, je vais me concentrer sur le code JS.

Il me faut :

Améliorer le code JS en mettant en place des fonctions de plus haut niveau.

⇆ Avec pour seule contrainte, de ne pas disposer de méthodes sur les tableaux.

L'objectif du projet est donc de proposer des fonctions capables de répondre à différentes contraintes.

Analyse et travail préparatoire

Nous allons prendre du recul et nous concentrer sur la structure de données de notre programme : le tableau d'objets.

objet

Intéressons nous aux objets : Les objets sont des collections de propriétés définies entre les {}. On parle de littéral objet.

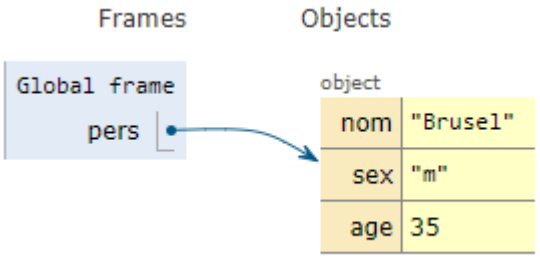
Soit un objet définit par trois propriétés :

```
{
  nom: "Brusel",
  sex : "m",
  age: 35
}
```

Il est facile de déclarer une personne

```
const pers =  {
  nom: "Brusel",
  sex : "m",
  age: 35
}
```

Voici sa représentation graphique.



tableau

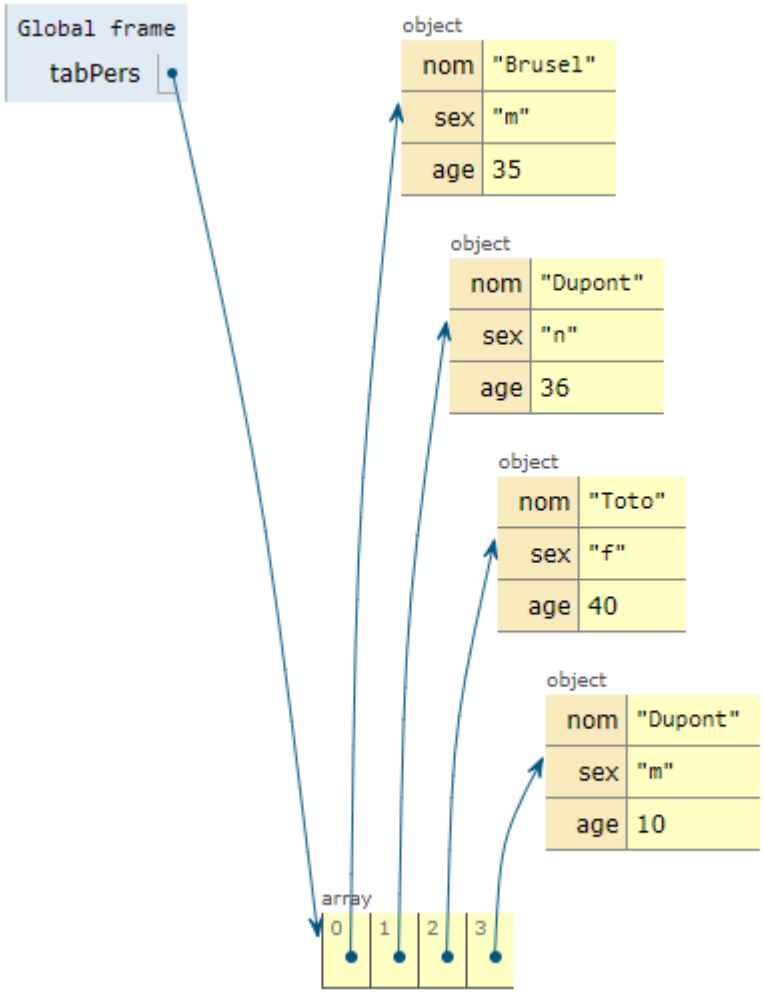
Intéressons nous maintenant au tableau : un tableau est un littéral défini entre [].

Un tableau de personnes peut alors se définir de la façon suivante.

```
1.  const tabPers = [
2.    {
3.      nom: "Brusel",
4.      sex : "m",
5.      age: 35
6.    },
7.    {
8.      nom: "Dupont",
9.      sex : "n",
10.     age: 36
11.   },
12.{
13.     nom: "Toto",
14.     sex : "f",
15.     age: 40
16.   },
17.   {
18.     nom: "Dupont",
19.     sex : "m",
20.     age: 10
21.   },
```

22.];

Voici la représentation du tableau sous sa forme graphique.



Nous pouvons accéder aux données de différentes façons :

tabPers[1]

-> {nom: "Dupont", sex: "n", age: 36}

L'accès à la valeur d'une propriété se fait de deux façons différentes.

1. tabPers[1]["nom"]
2. tabPers[1].nom

Si la première manière permet d'utiliser des expressions, la seconde est plus simple.

Les opérations de base

Nous allons maintenant commencer à écrire des opérations élémentaires sur le tableau. Nous allons principalement étudier les boucles sur un tableau.

👷 Ecrire la boucle *for* classique affichant les âges

"0 : 35 ans"
"1 : 36 ans"
"2 : 40 ans"
"3 : 43 ans"

Remplacer ? dans les codes suivants :

```
1. for (let i=0; i<  ? ; i++){
2.  console.log(`${i} : ?  ans`);
3. }
```

Nous pouvons utiliser, si nous n'avons aucun usage de l'index, la boucle ***for of***

👷 Réécrire la boucle avec *for of*

```
1. let i = 1;
2. for (let pers of tabPers) {
3.   console.log(`${i++} : ?  ans`);
4. }
```

👷 Utilisation la destructuration[5]

```
1. let i = 1;
2. for (let { ? } of tabPers) {
3.   console.log(`${i++} : ${a} ans`);
4. }
```

[5] https://dupontdistanciel.blogspot.com/2020/09/blog-post_54.html

tests

Nous allons commencer à répondre à une série de questions plus complexes.

Par exemple, j'aimerais obtenir un nouveau tableau contenant tous les hommes.

Dans le code suivant : lig.1, nous déclarons un nouveau tableau vide.

lig. 4 : Nous ajoutons un élément, pour cela nous disposons de la méthode push.

👷 Création d'un tableau des personnes de sex h

```
1. let tabPersF = [];
2. for (let { age, sex } of tabPers) {
3.     if ( ? ) {
4.         tabPersF.push( ?); // stocker l'âge
5.     }
6. }
7. console.log(tabPersF);
```

👷 Création d'un tableau des personnes de sex f

// inspirez vous du code précédent

👷 Passons maintenant à autre chose qu'un filtre sur le sex :

Ecrire le code pour obtenir les personnes

- les personnes majeures
- les personnes mineures
- les "Dupont"
- Les "Dupont" mineurs
- Les personnes dont le nom commence par un "A"
- Les personnes de sex feminun dont le nom commence par un "A" et qui son majeure.

- …

Bref, vous l'avez compris, le nombre de requêtes est infini.

⁉ Réfléchissez à la souplesse de votre code.

Quel effort vous demande la résolution d'une nouvelle interrogation.

Travail personnel

Compléter votre code sur jsbin.

https://jsbin.com/carumoc/1/edit?js,console

Nous reviendrons ensuite à notre projet initial qui affiche dans une page WEB les personnes en fonction de critères.

Pour aller plus loin

http://dupontquizz.blogspot.com/2017/11/ds-filtre-and-co.html