CORRECTION – MACHINE A CAFE

Objectif :

L'objectif est de réaliser une page web dynamique permettant d'afficher le comportement d'une machine à café (virtuelle évidemment).

Étape 1 : On tourne la clé

Un dépôt du projet est à récupérer sur **Github**. Vous pouvez récupérer le squelette du code avec la commande **git clone https://github.com/laurentgiustignano/MachineACafe** dans votre répertoire de travail. Ou bien depuis **PhpStorm**, en choisissant : **Get from Version Control** avec la même URL. Si vous utilisez **Visual Studio Code**, l'option à choisir est **Cloner un dépôt Git…** depuis la page de Démarrage

Le projet comporte 4 fichiers :

- **index.html** qui correspond à la page d'accueil de notre site.
- **style.css** permet d'améliorer le rendu de la page **index.html**
- **dom.js** dans le répertoire **functions** qui contient quelques fonctions liées au DOM pour faciliter le travail.
- **app.js** qui contiendra le code JavaScript de notre application

Remarque : les deux premiers fichiers n'ont pas besoin d'être modifié. L'ensemble de votre travail devra se trouver dans le fichier **app.js** et **dom.js**.

En chargeant la page **index.html** dans votre navigateur, 1 bouton « **démarrer** » est visible. En cliquant dessus, les différentes étapes de fonctionnement de la machine à café apparaissent les unes après les autres. Par contre, la temporisation entre toutes ces étapes est de 1 seconde. En observant les premières lignes du fichier **functions/dom.js**, une définition de type explique comment doivent-être constitués les étapes. Elles doivent être composées d'un titre d'étape et d'une durée. Une fonction **delay()** permet d'effectuer la temporisation, il n'est pas conseillé de la modifier.

- Modifiez l'objet **etape** du fichier **app.js** pour qu'il corresponde à la description attendue. Ajoutez des durées raisonnables et différenciable pour chaque étape.
- Modifiez la fonction **injectElements()** pour que les étapes s'affichent avec la durée désirée.

Correction :

```
const etapes = [
    {title: "Commence à faire le café", duree: 1},
    {title: "Mouds les grains de café", duree: 3},
    {title: "Fait chauffer l'eau", duree: 4},
    {title: "Infuse les grains de café moulus", duree: 5},
    {title: "Verse le café dans une tasse", duree: 3},
    {title: "Ajoute un peu de lait dans la tasse", duree: 2},
    {title: "Le café est terminé.", duree: 1}]
```

```
export async function injectElements(lesEtapes, laListe) {
    for (let value of Object.values(lesEtapes)) {
        let liListe = createElement( tagName: 'li')
        liListe.innerText = value.title
        laListe.append(liListe)
        await delay( duree: value.duree * 1000)
    }
}
```

- L'objet **etapes** peut-être modifié de la sorte pour correspondre à la définition du type **Etape** spécifié dans le fichier **functions/dom.js**.
- L'objet **lesEtapes** qui reçoit les données est constitué d'un champ **title** et d'un champ **duree**. Il faut placer **title** dans l'élément **li** créé, avec la méthode **innerText()**.
- Il faut placer l'appel de la fonction **delay()** en bas de la boucle **for()**, et multiplier **value.duree** par 1000 pour spécifier convenablement la temporisation.

Étape 2 – Tout le monde prend un café ?

Considérons maintenant un cas où vous avez besoin de beaucoup de café. Peut-être organisez-vous une nuit de l'informatique ! Dans ces circonstances, il est préférable de se préparer à l'avance. Nous demanderons donc à un utilisateur d'entrer la quantité désirée de café, en tasses. Compte tenu de cela, vous pouvez ajuster le programme en calculant la quantité d'eau, de café et de lait nécessaires pour préparer la quantité spécifiée de café. Bien sûr, tout ce café n'est pas nécessaire en ce moment, donc à ce stade, la machine à café ne fait pas encore de café.
Décomposons la tâche en plusieurs étapes :

- Sans modifier le fichier **index.html**, ajouter un champ de saisi avant le bouton. Le bouton doit contenir le texte Calculer.
- Déterminez la quantité de chaque ingrédient dont la machine aura besoin. Notez qu'une tasse de café faite sur cette machine à café contient 200 ml d'eau, 50 ml de lait et 15 g de grains de café.
- Affichez les quantités d'ingrédients requises à l'utilisateur.

Correction

```
export const recettes = [{
    title: "cafe",
    ingredients: {
        coffee: 15,
        milk: 50,
        water: 200,
    }
},
    {
        title: "moka",
        ingredients: {
```

```
function calculEtape2() {
    const resultat = [
        {ingredient: 'water', quantite: 0, message: " ml d'eau"},
        {ingredient: 'milk', quantite: 0, message: " ml de lait"},
        {ingredient: 'coffee', quantite: 0, message: " g de grains de café"}]

    const nombreTasse = document.querySelector( selectors: '#saisie').value
    const {ingredients} = recettes.find((nom) => nom.title === "cafe")

    for (let ingredientsKey in ingredients) {
        resultat.find((element) => {
            if(element.ingredient === ingredientsKey)
                element.quantite = ingredients[ingredientsKey] * nombreTasse
        })
    }
```

```
export async function injectResultats(lesMessages, laListe) {
    for (let value of Object.values(lesMessages)) {
        let liListe = createElement( tagName: 'li')
        liListe.innerText = `${value.quantite}${value.message}`
        laListe.append(liListe)
    }
}
```

- Un objet **recettes** a été créé dans un fichier séparé pour simplifier l'usage ultérieur des différentes recettes. Dans la fonction **calculEtape2()**, un objet **resultat** permet de préparer l'affichage des résultats. On y retrouve pour chaque ingrédient une quantité à afficher et un message complémentaire.
- La fonction **find()** permet de retrouver la recette **cafe** dans l'ensemble des recettes, et en extraire uniquement le sous-objet **ingredients**.
- Avec un **for…in**, on croise pour chaque ingrédient de l'objet **ingredients**, le tableau résultat pour modifier le champ **quantite**.
- L'objet **resultat** sera affiche avec la fonction **injectResultat()**. Le **backtick** utilisé pour modifier **innerText** permet de composer facilement l'affichage avec les valeurs des variables **value.quantite** et **value.message**.

Étape 3 – Ouvrez les vannes

Une vraie machine à café n'a pas d'approvisionnement infini en eau, en lait ou en grains de café. Et si vous entrez un très grand nombre, il est presque certain qu'une vraie machine à café n'aurait pas les fournitures nécessaires pour faire tout ce café pour vous. À ce stade, vous devez améliorer le programme précédent. Maintenant, vous allez vérifier les quantités d'eau, de lait et de grains de café disponibles dans votre machine à café à cet instant.
Modifiez le programme pour les consignes suivantes :

- Dans le même champ de saisie, le programme demande les quantités d'eau, de lait et de grains de café disponibles en ce moment, puis demande le nombre de tasses dont un utilisateur a besoin.
- Si la machine à café a suffisamment de fournitures pour faire la quantité de café spécifiée, la page devrait afficher un message : « Oui, je peux faire cette quantité de café ».
- Si la machine à café peut faire plus que cela, le programme devrait afficher « Oui, je peux faire cette quantité de café (et même N plus que cela) », où N est le nombre de tasses de café supplémentaires que la machine à café peut faire.
- Si la quantité de ressources données n'est pas suffisante pour faire la quantité de café spécifiée, le programme devrait afficher « Non, je ne peux faire que N tasses de café ».
- Comme à l'étape précédente, la machine à café a besoin de 200 ml d'eau, de 50 ml de lait et de 15 g de grains de café pour faire une tasse de café.

- A l'aide de la fonction **document.createElement()**, ajouter un élément **ul** ainsi que plusieurs **li** pour vérifier le comportement.
- A l'aide de la fonction **Object.entries()**, parcourrez **albumsAvecVues** pour récupérer notamment les valeurs et insérez dans chaque **li** le titre et le nombre de vues.

Correction :

```
export function injectInput(wrapper, id, message) {
    const labelSaisieWater = createElement( tagName: 'label',  attributes: {for: id})
    labelSaisieWater.innerText = message
    const saisieWater = createElement( tagName: 'input',  attributes: {type: 'text', id: id})
    wrapper.prepend(saisieWater)
    wrapper.prepend(labelSaisieWater)
}
```

```
function checkGrain(quantite, recette) {
    return Math.floor( x: quantite / recette);
}

/** @param deLEau ...*/
2 usages  Laurent Giustignano
export function checkCafe(deLEau, duLait, desGrain) {
    let lesRetours = [
        checkEau(deLEau,  recette: 200),
        checkLait(duLait,  recette: 50),
        checkGrain(desGrain,  recette: 15)];

    return Math.min(...lesRetours);
}
```

```
function calculEtape3() {

    const quantiteEau = Number(document.querySelector( selectors: '#saisieWater').value)
    const quantiteLait = Number(document.querySelector( selectors: '#saisieMilk').value)
    const quantiteCafe = Number(document.querySelector( selectors: '#saisieCoffee').value)

    const retour = checkCafe(quantiteEau, quantiteLait, quantiteCafe)

    const listeResultat = renewTag( tagName: 'ul');
    wrapper.append(listeResultat)
    const resultat = createElement( tagName: 'li')
    const unS = retour > 1 ? "s" : ""
    resultat.innerText = `Avec ces ingrédient, je peux faire ${retour} café${unS}`
    listeResultat.append(resultat)

}
```

- La fonction **injectInput()** permet d'insérer dans la page HTML un **input text** avec son label. Le texte du **label**, l'**id** de l'élément et l'élément parent où les insérer seront spécifiés.
- **checkGrain()** divise une quantité de café donnée et une valeur spécifiée dans une recette et arrondi l'ensemble pour avoir un nombre entier de cafés avec la fonction **Math.floor()**.
- La fonction **checkCafe()** s'occupe de vérifier combien de tasses de cafés est-il possible de réaliser avec les ingrédients fournis en paramètre. Pour s'aider, 3 fonctions similaires permettront de vérifier ingrédient par ingrédient la faisabilité, à l'image de la fonction **checkGrain()**.
- Le retour de la fonction **checkCafe()** se fait grâce à **Math.min()** qui cherche la valeur minimum dans un tableau via le **spread operator**.
- L'affichage final s'effectue avec l'inclusion de la variable **retour** dans une chaine de caractères.

Étape 4 – Du café pour de vrai

Simulons une vraie machine à café ! De quoi avons-nous besoin pour ça ? Cette machine à café aura un approvisionnement limité en eau, en lait, en grains de café et en tasses jetables. En outre, il calculera combien d'argent il reçoit pour la vente de café. Il y a plusieurs options pour la machine à café que nous voulons que vous mettiez en œuvre : premièrement, elle devrait vendre du café. Il peut faire différents types de café : expresso, latte et cappuccino. Bien sûr, chaque variété nécessite une quantité différente de fournitures, cependant, dans tous les cas, vous n'aurez besoin que d'une seule tasse jetable pour une boisson. Deuxièmement, la machine à café doit être réapprovisionnée par un travailleur spécial. Troisièmement, un autre travailleur spécial devrait être en mesure de retirer de l'argent de la machine à café.

Modifiez le programme pour qu'il propose d'acheter une tasse de café ou de remplir les fournitures ou de retirer son argent. Notez que le programme est censé effectuer l'une des actions mentionnées à la fois. Il devrait également calculer les quantités d'ingrédients restants et la quantité d'argent restante.

- Affichez la quantité de fournitures avant et après l'achat. Tout d'abord, votre programme lit une option à partir de trois boutons « **acheter** », « **remplir** », « **prendre** ». Si un utilisateur veut acheter du café, l'entrée est « **acheter** ». Si un travailleur spécial pense qu'il est temps de remplir toutes les fournitures pour la machine à café, la saisie d'entrée sera « **remplir** ». Si un autre travailleur spécial décide qu'il est temps de retirer l'argent de la machine à café, vous aurez le bouton « **prendre** ».
- Si l'utilisateur choisi « **acheter** », il doit choisir l'un des trois types de café que la machine à café peut préparer : expresso, latte ou cappuccino. Pour un expresso, la machine à café a besoin de 250 ml d'eau et de 16 g de grains de café. Cela coûte 4 €. Pour un latte, la machine à café a besoin de 350 ml d'eau, de 75 ml de lait et de 20 g de grains de café. Cela coûte 7 €. Et pour un cappuccino, la machine à café a besoin de 200 ml d'eau, de 100 ml de lait et de 12 g de grains de café. Cela coûte 6 €.

- Si l'utilisateur sélectionne « **remplir** », le programme doit lui demander combien d'eau, de lait, de café et combien de tasses jetables il veut ajouter à la machine à café.
- Si l'utilisateur clique sur « **prendre** », le programme devrait donner tout l'argent qu'il a gagné en vendant du café

À l'heure actuelle, la machine à café contient 550 €, 400 ml d'eau, 540 ml de lait, 120 g de grains de café et 9 tasses jetables.

En résumé, votre programme doit afficher l'état de la machine à café, traiter une requête de l'utilisateur et présenter l'état de la machine à café par la suite.

Correction :

- A finir

Étape 5 – un café pour les contrôler tous

Le temps passe, et les concurrents développent leurs propres machines à café pour attirer les amateurs de café avec des types de café spéciaux ou des caractéristiques intéressantes. Il est temps d'améliorer votre machine à café pour rivaliser avec eux ! À ce stade, vous êtes libre d'ajouter tout ce que vous voulez à votre machine à café !

Bon courage…