Le cours de JS va commencer !

Je vais laissé suffisamment de temps, aux débutants pour se mettre à niveau[1] sur les notions de base que sont :

- Types
- Opérateurs
- Structures de contrôle
- Fonctions

Je vous ai rédigé une intro tout en un !

Intéressons nous aux cours de ES6 et API DOM !

Cours ES6

Pour les notions de base, voici les liens

- Types : http://dupontes6.blogspot.com/p/type.html
- Opérateurs : http://dupontes6.blogspot.com/p/type.html
- Structures de contrôle : http://dupontes6.blogspot.com/p/4-structure-de-controle.html
- Fonctions : http://dupontes6.blogspot.com/p/les-fonctions-1.html

Des vidéos illutrent également certaines notions : https://dupontdistanciel.blogspot.com/2020/09/blog-post_54.html

Cours DOM

Il faudrait pour les grands débutants attaquer deux Montagnes que sont :

le DOM : https://dupontdom.blogspot.com/p/view-dom-svg.html

le CSS : https://dupontcss.blogspot.com/

On va donc tenter de faire du ES6, avec un minimum d'interface. C'est tout à fait possible et même c'est une bonne chose de penser que JS à quitter les navigateurs et leurs APIs.

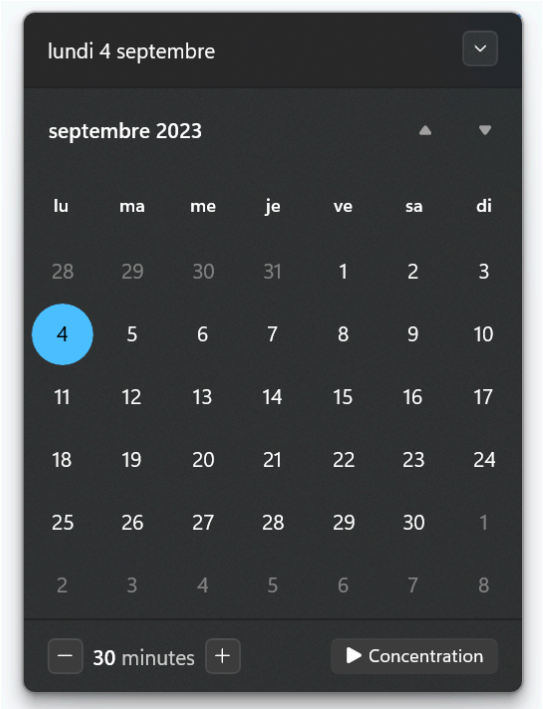
Je viens de créer un blog : https://dupontcourunpourcent.blogspot.com/ pour avoir en tête le vocabulaire de base.

[1] Vous aviez à loisir l'occasion de me contacter. Je reste dispo !

Bilan :

Le constat est clair. Au boulot.

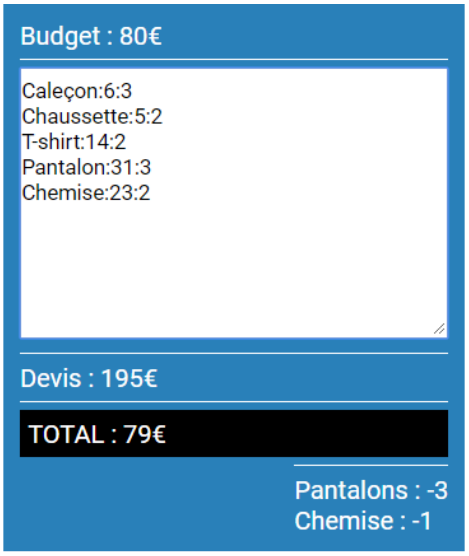
Et on va s'intéresser immédiatement à un projet : afficher un calendrier.



Des bibliothèques existent : exemple

TD, TP, Projets

Application des notions du cours à projet soldes.

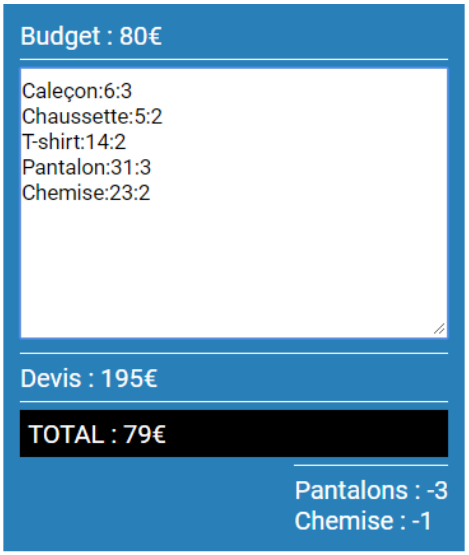


L'idée que j'ai pour vous, est de partir de notre code "débutant" et de l'améliorer vers un code plus "PRO".

Il ne resterait plus que l'ajout de la programmation objet (pour plus tard).

Travail préparatoire !

Reprenons le projet des soldes ! (enoncé)



Analyse du problème

Je n'ose vous demander :

"qui a rédigé un cahier des charges et a pris le temps d'écrire un pseudo-code ?".

C'est pourtant ce que vous devrez rédiger en tant que chef de projet Junior, laissant à des jeunes stagiaires le soin d'implémenter (en Java, …) une solution.

Choix d'une structure :

Et oui, vous le savez un algorithme traite une information structurée.

Mettre vos articles dans un tableau est un très bon choix.

Il s'agit principalement parcourir ensuite notre tableau[2].

Nous pourrions utiliser une liste chainée (ou doublement chaînée), je pencherais pour une structure **d'arbre équilibré**, facilitant la recherche du MAX (O(profondeur)[3]). Mais, je laisse au prof d'algo le choix de vous convaincre. Je ne vous apprends qu'à traduire vos choix en JS.

Solution de base

Prenons 5 minutes :

Je ne suis pas doué en rédaction. Lorsque j'écris "Prenons 5 minutes", je ne veux pas dire que j'ai mis 5 minutes pour implémenter mon proto.

J'ai mis en réalité plus d'un WE[4].

Non, l'idée de ce paragraphe est de vous inciter, une fois la mise en place d'un premier prototype, d'améliorer votre code *(en prenant 5 minutes de recul !)*

Analysons donc notre premier proto.

Analyse

Dans notre prototype de base, nous retirons à chaque itération l'achat le plus cher. C'est un choix[5] !

Mais, il y a un autre problème, celui des performances.

---

[2] help
[3] Complexité minimale
[4] Je pensais que le problème était facile à résoudre, je me suis emmêler les pieds dans des faux problèmes, par exemple créer une interface …
[5] 👷 Trouvez les achats à retirer pour s'approcher au mieux de notre budget est un problème théorique. Hors sujet !

Je vous ai donc demandé de chercher comment améliorer les performances du code de notre premier prototype ?

Voici deux axes d'améliorations.

Nous voyons lig. 54 que nous calculons pour chaque retrait le budget[6] :

```
54.  while (total(listeAchats) > budjet)
```

La seconde amélioration consisterait à être plus pertinent sur le retrait des articles.

Dans notre premier prototype, voici comment on retire un article :

1. *retire 1 article (ici pantalon) -> calculer le total*
2. *retire 1 article (pantalon) -> calculer le total*
3. *retire 1 article (pantalon) -> calculer le total*

On pourrait améliorer les performances en calculant directement la quantité de pantalon que l'on peut retirer sans pour autant calculer le total :

1. *retire 3 pantalons -> ~~calculer le~~ total-3*Prix*

Solution[7]

A partir de Lundi, nous allons améliorer notre style d'écriture !

Vous ne devrez plus écrire des boucles de transformation, des boucles pour tester des valeurs, des boucles pour faire un total.

Vous ne devez plus écrire de forEach également.

Très prochainement, vous transformerez, filtrez, réduirez[8] …

Vous appliquez vos connaissances sur le projet Soldes. Nous pourrons ainsi voir tous les bénéfices de ces nouvelles syntaxes.

Par la suite, nous verrons l'objet et nous finirons par un exemple de code pro (type MVC) d'un problème pédagogique très classique une TO-DO liste.

---

[6] Horreur !

[7] Il y a dans ce code de la déstructuration (une nouveauté)

[8] certains connaissent déjà ces méthodes.