# Javascript

Exercice asynchrone

# 1 - Mise en place

1. Créez deux fichiers:
   - async.html (avec une structure HTML de base)
   - async.js
2. Importez le fichier async.js à la fin du body du fichier async.html

# 2 - Définition des classes (1)

1. Définissez une classe ***Product*** qui possède:
   - Les attributs suivants: *id*, *name*, *stock*, *price* (id, stock et price seront des Number, name une String).
   - Un constructeur prenant 4 paramètres qui permettront d'affecter les 4 attributs correspondants.
   - Une méthode ***display*** qui retourne une chaîne de caractère contenant les attributs du produit.
   - Une méthode ***getHtml*** qui crée et retourne un élément HTML contenant les informations du produit (SANS insérer le produit dans le document via appendChild).
2. Définissez une classe ***Toy*** qui hérite de la classe ***Product*** et qui possède:
   - Un attribut *ageMin* (Number).
   - Un constructeur prenant 5 paramètres qui permettront d'affecter les 5 attributs correspondants.
   - Une méthode ***display*** qui retourne une chaîne de caractère contenant les attributs du jouet.
   - Une méthode ***getHtml*** qui crée et retourne un élément HTML contenant les informations du jouet (SANS insérer le jouet dans le document via appendChild).

# 2 - Définition des classes (2)

1. Définissez une classe ***Store*** qui possède:
   - Les attributs suivants: *address*, *productList* (address sera une String, productList un tableau de Product).
   - Un constructeur prenant 2 paramètres qui permettront d'affecter les 2 attributs correspondants.
   - Une méthode ***addProduct*** qui prend un Product en paramètre et l'ajoute au tableau productList.
   - Une méthode ***updateProductStock*** qui prend un identifiant et une quantité en paramètres, parcourt le tableau productList et ajoute la quantité au produit dont l'identifiant correspond à celui passé en paramètre.
   - Une méthode ***currentStockValue*** qui calcule et retourne la valeur totale du stock du magasin (sachant que valeur d'un produit = prix * quantité).
   - Une méthode ***fillHtml*** qui parcourt la liste des produits et, pour chaque produit, insère l'élément HTML dans le document (via appendChild). L'appel de cette méthode devra donc rendre la liste des produits visible dans la page HTML.
   - Une méthode ***needReorder*** qui retourne la liste des produits dont le stock est inférieur ou égal à 10 (vous pouvez par exemple créer un nouveau tableau vide, parcourir la liste des produits, ajouter les produits proche de la rupture de stock au nouveau tableau, puis le retourner).

# 3 - Utilisation des classes

1. Créez un magasin (Store).
2. Créez et ajoutez 2 produits (Product) et 2 jouets (Toy) au magasin. Vous veillerez à ce que chaque produit/jouet possède un id unique. Au moins un des produits devra avoir un stock inférieur à 10.
3. Affichez les produits proche de la rupture de stock (*needReorder*).
4. Affichez la valeur actuelle du stock du magasin (*currentStockValue*).
5. Modifiez le stock d'un produit (*updateProductStock*).
6. Affichez de nouveau la valeur actuelle du stock du magasin. Elle doit avoir changé.
7. Affichez les produits du magasin sur la page HTML (*fillHtml*)

# 4 - Pour finir...

1. Dans le document HTML, sur la ligne de chaque produit, ajoutez la valeur totale du stock pour ce produit (stock * prix)
2. Affichez les produits proches de la rupture de stock en rouge
3. Dans le document HTML, affichez la valeur totale du stock du magasin