algobase-training-course

Etape 3: Boucles

Typescript est compilé pour générer du Javascript, langage utilisé côté client pour les sites internet. Nous allons donc maintenant utiliser du TypeScript pour afficher des éléments dans une page web. Nous nous focaliserons sur l'affichage d'éléments graphiques dans une fenêtre (un objet de type *canvas* en HTML).

Ouvrez la page index.html du dossier algobase-training-course dans un navigateur. De la même façon que précédemment, ouvrez la console de développement web pour voir les éléments affichés et/ou les erreurs.

Dans la suite:

1. MODIFIER LE FICHIER moncode.ts (uniquement!!!!)
2. Compiler le code: > tsc moncode.ts
3. Actualiser la page index.html dans le navigateur (F5 ou bouton "Actualiser")
4. Voir le résultat

Copiez les codes dans un autre fichier si vous souhaitez les conserver.

Question 1: un rectangle

Ecrivez le code suivant:

```
dessinerRectangleRempli(10,10,30,30,"blue");
```

Compilez et actualisez la page.

Ajoutez le code à la suite pour tracer 4 autres rectangles, de taille 30x30, remplis avec des couleurs différentes, à des positions différentes.

Question 2

Modifiez le code de façon à tracer les 5 rectangles de la même couleur (utilisez une nouvelle variable appelée *couleur*), avec posY=200 et posX commençant à 0 puis ensuite 100 pour le 2ème rectangle, 200 pour le 3ème, ...

Problèmes d'itérations

Si vous avez ajouté une instruction par rectangle, vous devez vous rendre compte que vous avez répéter plusieurs fois la même instruction en faisant varier la position en X du rectangle.

répétition => boucle

Nous répétons une instruction similaire 5 fois. Nous pouvons donc employer la boucle **Pour (for).** Cette boucle nécessite:

- une variable entière dont la valeur sera modifiée itérativement. Ici, nous codons la même instruction en faisant varier la position en X du rectangle => nous utiliserons posX:number (rappelez-vous, il faut utiliser des noms de variables explicites)
- une condition de départ: quelle est la valeur initiale de notre variable => posX=0
- une condition finale: la position doit être inférieure ou égale à 400 => posX<=400
- le "comportement" de notre variable entre chaque itération: ici, nous devons ajouter 100 à la position en X => posX=posX+100 Un raccourci d'écriture que l'on retrouve dans de nombreux langages de programmation est possible: posX +=100; Pour un incrément de 1: posX++;

Voici le code pour notre boucle:

```
let posX:number;
for(posX=0;posX<=400;posX=posX+100){
    // l'instruction à répéter (ici: dessiner un rectangle en posX,200)
}
```

Question 3

Remplacez vos instructions par le contenu de la boucle. Testez.

Amélioration

Un inconvénient majeur du code proposé précédemment est que l'on ne retrouve pas notre contrainte: "nous voulons 5 rectangles" dans notre code. Il faudrait avoir dans notre boucle la condition de fin: <= 5 ou, pour aller plus loin encore, <= nbrectangles avec la variable nbrectangles initialisée au début du code. On pourrait alors facilement changer le nombre de rectangles que l'on veut.

Nous allons utiliser dans la boucle:

- une variable entière dont la valeur sera le numéro du rectangle => nous utiliserons norectangle:number
- la condition de départ: norectangle=0 La position en X où placer le rectangle norectangle est donc: 100*norectangle
- une condition finale: norectangle < nbrectangles (comme nous avons commencé à 0, il faut aller jusqu'à nbrectangles-1 pour obtenir nbrectangles rectangles)
- le "comportement" de notre variable entre chaque itération: norectangle++

-

```
let nbrectangles:number=5;
let norectangle:number=0;
for(norectangle=0;norectangle<nbrectangles;norectangle++){
    dessinerRectangleRempli(100*norectangle,200,10,100,"blue");
}
```

i Il ne faut pas oublier de définir et initialiser la variable nbrectangles **!**

Question 4

Copiez ce code, compilez, testez.

Question 5

Modifiez le code pour afficher `8` rectangles. Compilez, testez.