algobase-training-course

Etape 4: Boucles, entrées et conditions

Question 1: saisie de l'utilisateur

Reprenons notre code précédent:

```
/// <reference path="fonctions.ts"/>
creerZoneDessin();
let nbrectangles:number=5;
let norectangle:number=0;
for(norectangle=0;norectangle<nbrectangles;norectangle++){
    dessinerRectangleRempli(100*norectangle,200,10,100,"blue");
}
```

Un des avantages d'avoir utilisé une boucle est que l'on peut (ici) afficher autant de rectangles que l'on veut, sans avoir à ré-écrire complètement notre code. Pour aller plus loin: nous allons demander au visiteur du site combien de rectangles il souhaite afficher. Nous utiliserons la fonction lireNombre. Celle-ci prend comme paramètre le message à afficher à l'utilisateur (une chaîne de caractères donc) et retourne le nombre saisi par l'utilisateur. Au lieu de

```
let nbrectangles:number=5;
```

indiquez

```
let nbrectangles:number=lireNombre("Combien de rectangles voulez-vous aff
```

Question 2: problème de place

En pratique, l'utilisateur peut rentrer le nombre qu'il veut. Il ne faut pas dessiner des rectangles en dehors de la zone de dessin. Nous allons donc:

1) calculer combien de rectangles rentrent dans la fenêtre (variable nbrectangles_max). Ce nombre vaut la largeur de la fenêtre (que l'on obtient avec la fonction largeurZone()) divisé par l'espace entre chaque rectangle (100).

2) nous ne dessinerons que les rectangles qui rentrent dans la fenêtre. Une solution consiste à ne pas executer l'instruction dessinerRectangleRempli si nbrectangles > nbrectangles_max. Cependant, la boucle est toujours parcourue (pour rien). Une alternative est donc de modifier le nombre de rectangles à afficher: soit c'est le nombre rentré par l'utilisateur soit c'est le nombre maximum de rectangles. Nous

allons donc modifier la valeur de nbrectangles avant de rentrer dans la boucle, à l'aide d'un test conditionnel (SI).

```
if(nbrectangles>nbrectangles_max)
{
    console.log("Valeur trop grande, nous ne tracerons que "+nbrectangles
    nbrectangles=nbrectangles_max;
}
```

i Le **+** correspond à la concaténation de chaînes de caractères

i A-t-on besoin de faire des tests si l'utilisateur rentre un nombre négatif ou nul **?** Non car nous ne rentrerons pas dans la boucle POUR **!**

Question 3: changement de couleur

Nous allons modifier le code de façon à ce que la couleur du rectangle soit choisie en fonction de la valeur saisie par l'utilisateur: quand le nombre est pair, les rectangles seront en bleu, en rouge dans l'autre cas. Définissez une variable de type string nommée couleur. Modifiez la ligne dessinerRectangleRempli en mettant couleur à la place de "blue". L'instruction pour assigner la couleur bleu à couleur est:

```
couleur="blue";
```

Cette instruction n'est pas toujours utilisée **=>** il faut utiliser un SI (test conditionnel). En Typescript: SI -> if

```
if(nbrectangles%2 == 0) {
    couleur="blue";
}
```

i **%** correspond au modulo, c'est à dire le reste de la division entière par le nombre qui suit. Les nombres pairs sont des multiples de **2** donc il ne reste rien de la division par **2**.

On n'a que deux situations: les pairs (bleu) et les impairs (rouge) donc l'instruction pour assigner la couleur rouge est utilisée dans le SINON

```
if(nbrectangles%2 == 0) {
    couleur="blue";
}
else {
    couleur="red";
}
```

Codez et testez.

Entrez la valeur 15. Est-ce que les rectangles sont en rouge ? Si ce n'est pas le cas: où avez-vous mis le SI ... SINON ? Si vous l'avez mis après le SI de la Question 2, l'erreur est là: vous choisissez la couleur après avoir modifié la valeur du nombre de rectangles à tracer (si cela dépassait de la fenêtre). Mettez le code entre la partie correspondant à la saisie par l'utilisateur et le SI de la question 2. Testez.

Question 4: changement de couleur (2)

Nous allons modifier le code: un seul rectangle demandé -> rose, deux rectangles -> vert, 3 rectangles -> rouge et, dans les autres cas, bleu. La couleur dépend toujours du nombre saisi par l'utilisateur mais il n'est plus question de pair ou impair. Comme auparavent, l'instruction couleur="pink" (couleur="green", couleur="red", ...) n'est pas toujours utilisé => SI. On a ici plusieurs cas possible => SI ... SINON SI ... SINON SI .... SINON

```
if(nbrectangles == 1) {
    couleur="pink";
}
else if(nbrectangles == 2) {
    couleur="green";
}
else if(nbrectangles == 3) {
    couleur="red";
}
else {
    couleur="blue";
}
```

Codez et testez.

Question 5: changement de couleur (3)

Nous changeons d'avis: la couleur des rectangles ne dépendra plus de ce qui est saisi par l'utilisateur mais de la position du rectangle: le 1er rectangle est rose, le deuxième vert, le troisième rouge et les autres bleus. La couleur va donc varier pour chaque rectangle: les tests condtionnels seront donc dans la boucle POUR et le test dépend de norectangle !

```
for(norectangle=0;norectangle<nbrectangles;norectangle++){
    if(norectangle == 0) {
        couleur="pink";
    }
    else if(norectangle == 1) {
        couleur="green";
    }
    else if(norectangle == 2) {
```

```
        couleur="red";
    }
    else {
        couleur="blue";
    }
    dessinerRectangleRempli(100*norectangle,200,10,100,couleur);
}
```

Remarque: la variable couleur est maintenant utilisée uniquement dans la boucle. On peut donc la définir dans la boucle

```
for(norectangle=0;norectangle<nbrectangles;norectangle++){
    let couleur:string;
    if(norectangle == 0) {
```