

TP 8

JavaScript : dessin et formulaires

Ce TP est dédié à JavaScript. Sauf mention contraire, aucune modification des fichiers HTML fournis n'est autorisée.



EXERCICE

Exercice 8.1 (Dessin animé). L'objectif est de réaliser (en JavaScript) un dessin similaire à celui de la [figure 8.1a](#).



Question 1 : Télécharger le fichier [dessin.html](#). Créer un fichier [dessin.js](#). On ne travaillera que dans ce fichier JavaScript.

Implémenter une méthode `dessine()` qui dessine la forme de la [figure 8.1a](#) grâce à JavaScript. Quelques fonctions utiles : `fillStyle`, `fillRect`, `beginPath`, `moveTo`, `closePath`, `lineTo`, `arc...` (Voir le cours.)

Le carré noir fait 500 px de côté (on pourra utiliser une constante).

Comme d'habitude, appeler cette méthode dans un auditeur d'évènement sur l'évènement `'load'`, par exemple :

```
1 window.addEventListener("load", () => {  
2   dessine();  
3 }
```



Question 2 : Ajouter des gestionnaires d'évènement pour que :

- un clic de souris décale l'arc de cercle bleu de $\frac{\pi}{8}$ dans le sens trigonométrique (anti-horaire);
- la pression de la touche l décale l'arc de cercle bleu de $\frac{\pi}{8}$ dans le sens trigonométrique;
- la pression de la touche d décale l'arc de cercle bleu de $\frac{\pi}{8}$ dans le sens antitrigonométrique (sens horaire).

Indications : ajouter une variable globale `theta` qui encode l'angle initial de l'arc. Puis implémenter deux méthodes `decaleAntiTrigo()` et `decaleTrigo()` qui changent la valeur de `theta`, puis appellent `dessine`. Enfin, ajouter dans les auditeurs concernés les appels à ces deux fonctions, par exemple :

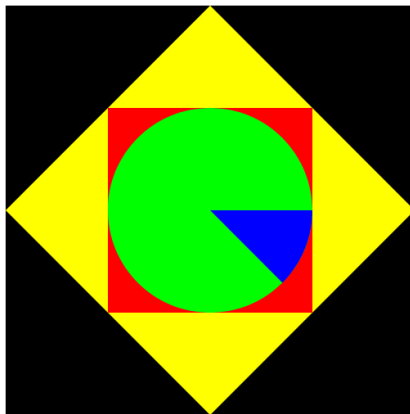
```
1 document.addEventListener('keypress', (e) => {if (e.key === 'g'){  
   decaleTrigo(); }});
```

Un exemple de rendu après 3 clics de souris est donné sur la [figure 8.1b](#).



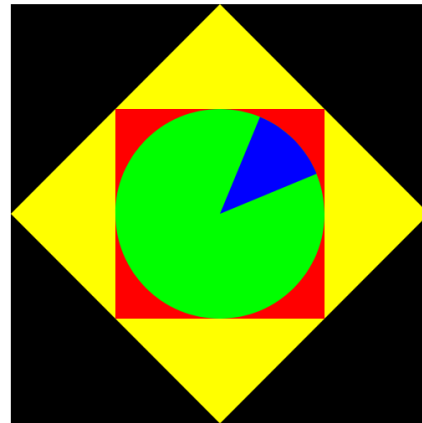
Question (optionnelle) 3 : Ajouter un contrôle par les flèches du clavier, en plus des touches l et g.

JavaScript : canevas et dessin




(a) Dessin initial

JavaScript : canevas et dessin



(b) Dessin après 3 clics

FIGURE 8.1 – Dessin

 **Question (optionnelle) 4 :** Ajouter deux boutons radio « Sens horaire » et « Sens antihoraire » dans la page HTML (qu’il est autorisé de modifier). (Aucun bouton « envoyer » n’est demandé, ni même un élément `<form>`.)


En fonction de la sélection du bouton, les clics de souris suivants doivent occasionner une rotation horaire ou antihoraire.

■




EXERCICE


Exercice 8.2 (Aleart).


 **Question 1 :** Créer un fichier HTML `aleart.html`, contenant un simple canevas de dimensions 400×400 , et le lier à un fichier `aleart.js`. Écrire une fonction `aleArt` qui affiche 500 rectangles de tailles et couleurs aléatoires (`r, g, b` et transparence) à une position aléatoire dans le canevas. À chaque clic de l’internaute sur le canvas, la fonction est à nouveau appelée, et le dessin est entièrement régénéré.

Un exemple de rendu est donné sur la [figure 8.2a](#) (ne pas tenir compte du formulaire à ce stade).

 **Question (optionnelle) 2 :** Ajouter à la page HTML un champ de formulaire pour paramétrer le nombre de rectangles. Lors de l’appui sur un bouton dédié, la page se régénère avec le nombre de rectangles entré.

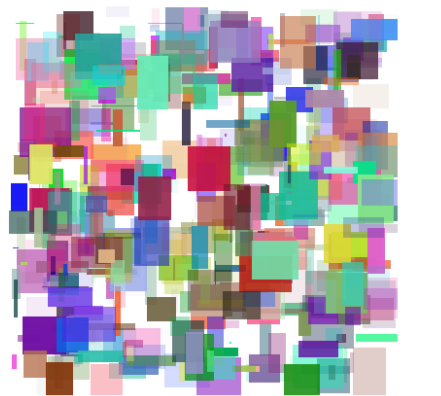
Un exemple de rendu est donné sur la [figure 8.2b](#).

 **Question (optionnelle) 3 :** Ajouter à la page HTML un menu déroulant permettant de choisir entre des ellipses ou des rectangles. Lors de l’appui sur un bouton dédié, la page se régénère avec le nombre de formes entré.

 **Question (optionnelle) 4 :** Ajouter un formulaire permettant à l’internaute de choisir la taille du *canevas* de façon dynamique.

■

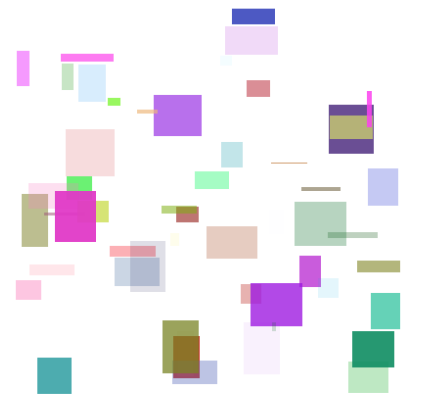
AleArt



Nombre de figures : Dessiner

(a) 500 rectangles aléatoires

AleArt



Nombre de figures : Dessiner

(b) 54 rectangles aléatoires

FIGURE 8.2 – Dessin



EXERCICE

Exercice 8.3 (Emojis (exercice pour les plus rapides)).

Question (optionnelle) 1 : Créer une page HTML [emojis.html](#) avec un simple formulaire, et deux boutons « + » et « - ». Voir rendu sur la [figure 8.3a](#) (sans les emojis, pour l’instant).

Question (optionnelle) 2 : Lier la page HTML à un fichier JavaScript [emojis.js](#) qui doit permettre de générer n emojis aléatoires, où n est la valeur du champ. Il doit être possible de modifier n en appuyant sur l’un des deux boutons (auquel cas la valeur du champ progresse elle aussi), ou en entrant une valeur directement dans le champ puis en appuyant sur la touche « entrée ».

Attention : les emojis ne doivent pas tous être régénérés à chaque modification du champ. Par exemple, s’il y a 20 emojis et que l’on entre 13 dans le champ, 7 emojis doivent être enlevés, mais les 13 premiers sont conservés.

Exemple d’utilisation :

1. initialement, le champ vaut 0;
2. on appuie ensuite 5 fois sur le bouton « + » : un résultat possible est donné sur la [figure 8.3a](#);
3. on entre ensuite « 20 » dans le champ, et on appuie sur la touche « entrée » : un résultat possible est donné sur la [figure 8.3b](#) (noter que les 5 premiers emojis ont été conservés);
4. on entre ensuite « 13 » dans le champ, et on appuie sur la touche « entrée » : un résultat possible est donné sur la [figure 8.3c](#) (noter que les 13 premiers emojis ont été conservés).

Indications :

- maintenir un tableau (constant) `TOUS_EMOJIS` d’une sélection d’une dizaine d’emojis de votre choix, parmi lesquelles on choisira dans ce qui suit les emojis aléatoires;
- maintenir deux variables globales : la valeur du champ (initialement 0), et le tableau courant `tableau_emojis` des emojis (initialement vide);
- créer une fonction `actualise_tableau()` qui actualise le tableau `tableau_emojis` en fonction de sa taille courante, et de la valeur du champ; si la taille courante est supérieure à la valeur, supprimer suffisamment d’éléments de `tableau_emojis`; sinon, en

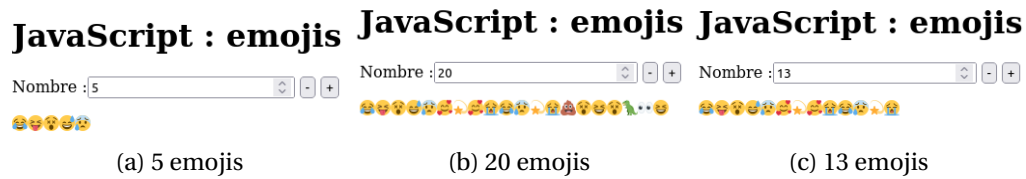



FIGURE 8.3 – Emojis aléatoires

ajouter suffisamment, en piochant à chaque fois un élément de façon aléatoire dans `TOUS_EMOJIS`;

- créer une fonction `regenerer_emojis()` qui régénère dans la page HTML le paragraphe contenant les émojis à partir de `tableau_emojis` (on pourra utiliser `tableau_emojis.join('')`);
- ajouter des auditeurs sur les boutons « + » et « - », qui vont simplement actualiser la valeur du champ, puis appeler `actualiser_tableau()` et enfin `regenerer_emojis()`;
- ajouter un auditeur sur le champ, qui se déclenche sur la touche « entrée », et fait les actualisations nécessaires à partir des fonctions prédéfinies. Pour vérifier que la chaîne entrée est bien un nombre, on peut utiliser :

```

1 function est_un_nombre(chaine){
2   return /^\d+$/ .test(chaine);
3 }
    
```

 **Question (optionnelle) 3 :** S’assurer que le bouton « - » est désactivé si la valeur du champ est < 1.

