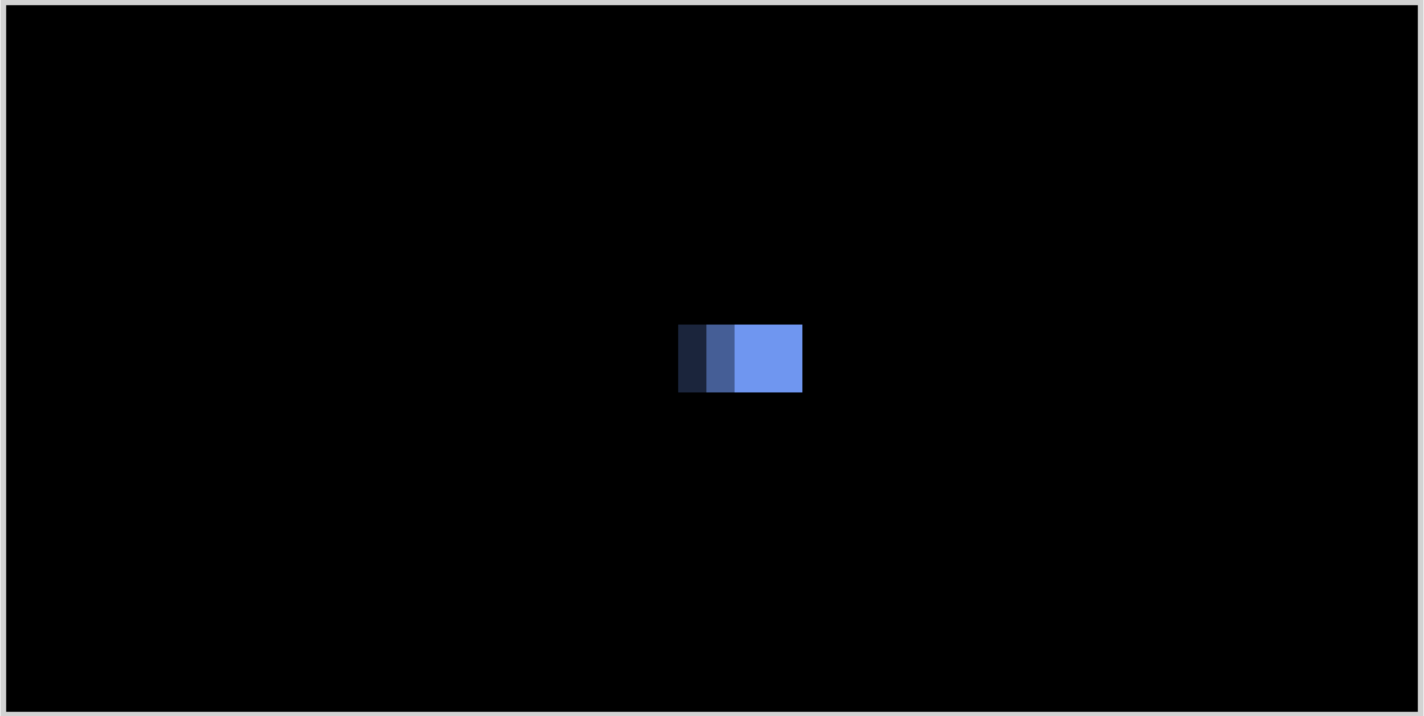
Déplacer un carré

Écrire une page Web sur laquelle on peut déplacer un carré sur un canevas à l'aide des touches du clavier.



ÉTAPE 1 – Création du canevas

Créer d'abord la page HTML5 et ajouter un élément canevas (`<canvas>`). Ne pas oublier de donner une taille au canevas à l'aide des attributs `width` et `height` de l'élément HTML.

Identifier le canevas (ex. : `id="canevas-jeu"`) pour qu'il soit possible d'y appliquer un style avec CSS et de récupérer l'élément dans le code JavaScript.

Finalement, personnaliser le canevas à l'aide de CSS en lui donnant une bordure et une couleur d'arrière-plan.

ÉTAPE 2 – Affichage du carré dans une boucle de jeu

Lier un script JavaScript à la page : `<script src="js/déplacer_carre.js"></script>`.

Dans ce script, construire une boucle de jeu de base :

```
window.onload = function () {

    canevas = document.getElementById('canevas-jeu');
    contexte = canevas.getContext('2d');

    window.requestAnimationFrame(boucleJeu);  // le navigateur appellera boucleJeu() au bon moment
}

function boucleJeu(timeStamp){

    // le code pour les traitements viendra ici

    // le code pour l'affichage viendra ici

    window.requestAnimationFrame(boucleJeu);  // le navigateur appellera boucleJeu() au bon moment
}
```

Choisir une couleur pour le carré. Affecter cette couleur au contexte (`contexte.fillStyle`) dans la fonction qui est appelée au chargement de la fenêtre (la fonction associée à `window.onload`). Pourquoi on place ce code à cet endroit? Pour ne pas l'exécuter à chaque passage dans la boucle de jeu : on garde le même style de remplissage pour chaque affichage du carré, il n'est pas nécessaire de le choisir à chaque fois.

Déclarer deux variables globales (au début du fichier JavaScript, avant toutes les fonctions) : `posX` et `posY`. Ces deux variables contiendront la position du carré dans le canevas. C'est cette position qui sera changée pour déplacer le carré.

Définir une constante pour la dimension du carré (ex.: `const DIMENSION_CARRE = 48;`). Cette constante doit aussi être globale. Elle permettra de préciser la position du carré et de limiter les déplacements à la surface du canevas. En effet, on ne veut pas que le carré s'évade du canevas!

Au chargement de la fenêtre (donc dans la fonction associée à `window.onload`), centrer le carré sur le canevas. La position X (variable `posX`) centrée sera le résultat d'un calcul impliquant la largeur du canevas (`canevas.width`) et la largeur du carré. La position Y (variable `posY`) centrée du carré sera calculée à partir de la hauteur du canevas (`canevas.height`) et de la hauteur du carré.

Ajouter une **fonction d'affichage** qui efface le canevas dans un premier temps (méthode `clearRect()` du contexte, ex. : `contexte.clearRect(…)`), puis qui dessine le carré plein à la bonne position sur le canevas (méthode `fillRect(…)`). Appeler cette fonction dans la boucle de jeu.

Lorsque ça fonctionne, siffloter « *It's a Small World* » des frères Sherman en agitant les bras pour aviser l'enseignant qui viendra vérifier le travail.

ÉTAPE 3 – Détection des touches du clavier

Lorsque l'utilisateur appuie sur une touche du clavier, un événement **keydown** est produit. Lorsque l'utilisateur relâche une touche du clavier, un événement **keyup** est produit. Dans les deux cas, le code de la touche appuyée ou relâchée est stocké dans la propriété **key** de l'événement. L'événement en question est un objet qui est passé en paramètre à la fonction responsable de la gestion de l'événement (*event handler*). Cette fonction responsable de la gestion de l'événement, on l'associe à un « écouteur d'événement » (*event listener*).

Créer tout d'abord deux fonctions de gestion des événements, par exemple :

```
function toucheAppuyee(evenement) {
    // le code de gestion de l'événement viendra ici
}

function toucheRelachee(evenement) {
    // le code de gestion de l'événement viendra ici
}
```

Ensuite, associer ces fonctions aux événements au début de la fonction associée au chargement de la fenêtre :

```
window.onload = function () {

    window.addEventListener("keydown", toucheAppuyee);
    window.addEventListener("keyup", toucheRelachee);

    ...
}
```

Vérifier que les deux fonctions sont bien appelées en insérant temporairement des affichages à la console (ex. : `console.log("touche appuyée: " + evenement.key);`). Il sera alors de vérifier les appels à l'aide des outils de développement du navigateur. Prendre en note les différentes valeurs de la propriété **key** pour les quatre touches du clavier : flèche vers le haut, flèche vers le bas, flèche vers la gauche et flèche vers la droite.

Il faudra stocker l'état des quatre touches qui nous intéressent de telle sorte qu'on puisse vérifier, le moment venu de déplacer le carré, si une touche en particulier est enfoncée ou pas. Puisque seules deux valeurs nous intéressent (touche enfoncée ou pas), on peut utiliser une valeur booléenne pour représenter cette information : `true` si la touche est enfoncée, `false` si elle ne l'est pas.

Créer un objet global (au début du fichier JavaScript, avant les fonctions) qui servira à stocker l'état de chacune des quatre touches. L'objet contiendra donc quatre (4) combinaisons « **clé: valeur** » où la clé sera la chaîne de caractères associée une touche (les chaînes de caractères que vous avez notées) et la valeur booléenne initialement fausse (`false`).

Finalement, insérer le code de gestion requis dans chacune des deux fonctions de gestion (*event handlers*) pour assigner la bonne valeur à la bonne clé selon la touche qui est enfoncée ou relâchée.

Lorsque c'est complété, siffloter « *Don't Worry, Be Happy* » du chanteur américain Bobby McFerrin en agitant les bras pour aviser l'enseignant qui viendra vérifier le travail.

ÉTAPE 4 – Ajustement de la position du carré

Avant d'afficher le carré, on doit maintenant calculer sa nouvelle position en fonction de l'état des touches. C'est le principal traitement à réaliser dans cette application : Si la flèche vers la gauche est enfoncée, on déplace le carré vers la gauche. Si la flèche vers la droite est enfoncée, on déplace le carré vers la droite, etc.

Pour y arriver, créer une fonction dont le rôle est de calculer la position du carré. Par exemple :

```
function calculerPosition() {
    // le code de calcul de la position viendra ici
}
```

Dans cette fonction, vérifier l'état de chacune des touches (on se souvient que les états sont stockés dans un objet global – voir étape précédente). Ensuite, en fonction de l'état, ajuster la position horizontale (variable globale `posX`) et la position verticale (variable globale `posY`) du carré.

Dans la boucle de jeu, appeler la fonction de calcul de la position juste avant l'affichage du carré. Recharger la page et tester les déplacements.

Lorsque c'est complété, yodler « *Hocus Pocus* » du groupe néerlandais Focus en agitant les bras pour aviser l'enseignant qui viendra vérifier le travail.

ÉTAPE 5 – Emprisonnement du carré dans le canevas

Pour emprisonner le carré dans le canevas, il suffit de modifier les résultats du calcul de position. Si le résultat est hors limite, on assigne la limite comme nouvelle position. Par exemple, si on obtient une valeur négative pour `posX`, on lui assigne plutôt la valeur 0. Même chose pour `posY`. Dans le cas des limites à droite et en bas, il faut plutôt utiliser la largeur ou la hauteur du canevas à laquelle on soustrait la largeur ou la hauteur du carré.

Lorsque c'est complété, aviser l'enseignant pour qu'il puisse célébrer lui aussi!