



JavaScript & Node.js

Exercices

1 Le DOM et la gestion d'événements : Animation via l'API Canvas

1.1 Animation de carrés à l'aide de l'API Canvas

Veillez adapter une application web permettant l'affichage de différents scintillements de formes selon la demande de l'utilisateur.

Le code à adapter se trouve dans le dossier [/demo/canvas](#) du repo associé au cours.

Veillez retravailler l'animation pour que :

- elle s'arrête ou démarre lors d'un click d'un utilisateur ;
- les carrés grandissent ou rapetissent au clic sur la touche + ou la touche - ;
- lors d'un clic droit de la souris, la couleur change de manière aléatoire.



- Vous pouvez utiliser le type d'événement « *contextmenu* » pour gérer les clics droits.

Plus d'info : https://developer.mozilla.org/en-US/docs/Web/API/Element/contextmenu_event

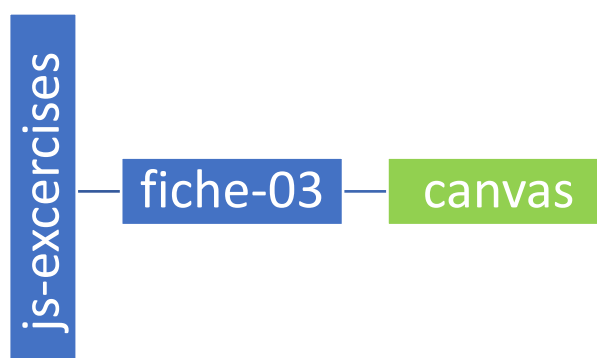
- Vous pouvez utiliser le type d'événement « *keydown* » pour gérer l'appui sur les touches + ou -

Plus d'info : https://developer.mozilla.org/en-US/docs/Web/API/Document/keydown_event

- Si vous avez besoin de générer un nombre aléatoire de 0 à 255 :

```
Math.floor(Math.random() * 256); // [0,255]
```

Le code de votre application web doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-exercices**).





JavaScript & Node.js

Exercices

2 Les modules (ES6) & structurer son code en orienté-objet

2.1 Gestion locale d'une collection de films

Veillez continuer le développement de **myMoovies** afin de permettre d'enregistrer temporairement et localement des films et d'afficher les films enregistrés en dessous du formulaire.

Votre formulaire doit contenir les champs :

- **title** : titre du film
- **duration** : durée du film en minutes
- **budget** : pour informer du coût qu'a coûté la production du film, en millions
- **link** : pour donner un lien vers la description du film (lien vers imdb, rottentomatoes ou autre)

A chaque submit du formulaire, vous allez afficher la liste des films « enregistrés ». Voici un exemple d'affichage :

Title	Duration (min)	Budget (million)
Harry Potter and the Philosopher's Stone	152	125
Avengers: Endgame	181	181
...		

Veillez noter que le champs **link** est utilisé pour créer un lien dans la colonne **Title**.

Veillez valider votre formulaire.

Afin de réaliser cet exercice, voici les contraintes d'implémentation :

- Créez une classe **FilmLibrary** au sein du module **FilmLibrary.js**
- Créez une classe **Film** au sein du module **Film.js**
- Le module **FilmLibrary.js** et le module **Film.js** seront appelés par votre script d'entrée **index.js**
- La classe **FilmLibrary** doit fournir comme méthode :
 - o **addFilm(film)** : ajoute un film dans une structure de données, un **Array** d'instances de **Film** ;
c'est cette méthode qui doit être appelée par votre script d'entrée



JavaScript & Node.js

Exercices

- ([index.js](#)) pour « enregistrer » un film au sein d'une structure de données après chaque submit valide de formulaire ;
- [getHtmlTable\(\)](#) : renvoie une string contenant la représentation HTML de la structure de données (un [Array](#) d'instances de [Film](#)) ; c'est cette méthode qui doit être appelée par votre script d'entrée ([index.js](#)) pour afficher la liste des films enregistrés, après chaque submit valide de formulaire ;
- Veuillez repartir du code associé à votre dernier exercice sur [myMoovies](#) : [/js-exercises/fiche-02/mymoovies](#). Dans un premier temps, n'hésitez pas à simplement compléter la Homepage.
Il n'est pas nécessaire de créer de nouvelle page ni de composant de navigation pour traiter de l'ajout d'un film et de la vue de la liste des films ajoutés.

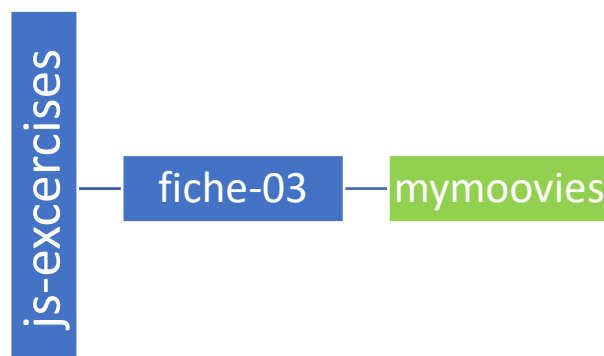


- Comme vous l'auriez fait dans d'autres exercices, vous pouvez utiliser la propriété [.innerHTML](#) d'une div pour afficher dynamiquement la table.
- Dans le cadre de l'exercice « Création d'une table sur base d'un formulaire » ([/js-exercises/fiche-02/dynamic-html](#)) de la fiche 02, vous avez créé une fonction qui renvoie sous forme de string une table HTML basée sur un [Array](#). N'hésitez pas à vous en inspirer pour la méthode [getHtmlTable\(\)](#).



Veuillez faire un [commit](#) de votre code avec le message suivant :
« myMoovies : step 3 : refactoring with modules & classes. New : add a film and view all films ».

Le code de votre application web doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository ([js-exercises](#)).





3 Exercices optionnels

3.1 Refactor de code en modules et gestion dynamique d'HTML

3.1.1 Refactor de la gestion locale d'une collection de films

N'hésitez pas, c'est optionnel, de créer deux pages pour votre application **myMoovies** ainsi qu'un composant de navigation.

La première page serait la Homepage telle que développée dans le cadre de la fiche 02. La deuxième page reprendrait les fonctionnalités d'ajout d'un film et d'affichage des films telles que décrites à l'exercice précédent (§2.1).

Afin de réaliser cet exercice optionnel, voici les contraintes d'implémentation :

- **index.js** doit faire appel soit au module **Homepage**, soit au module **Filmpage**, en fonction des clic sur le composant de **Navigation**.
- La gestion des clics sur le composant de **Navigation** doit être gérée dans le module **Navigation**.



Veuillez faire un **commit** de votre code avec le message suivant :
« myMoovies : step 3+ : refactoring to create 2 pages. New : navigation ».

3.2 Animation via Anime.js

3.2.1 Animation d'un nuage de mots à l'aide d'Anime.js

Vous avez la possibilité, en supplément, de réaliser une application web permettant d'afficher un nuage de mots en mouvement.

Au passage de la souris sur un mot, il doit s'agrandir.

Au clic sur un mot :

- faites disparaître tous les autres mots ;
- affichez d'une manière élégante la description associée au mot qui a été cliqué.

Afin de réaliser cet exercice nous vous proposons ces contraintes d'implémentation :



Exercices

- Utilisez cet **Array** pour créer vos mots de manière dynamique (utilisez soit la propriété **title** ou l'**url**)

```
const animationLibraries = [  
  {  
    title: "Anime.js",  
    url: "https://animejs.com/",  
    description: `Anime.js (/ˈæn.ə.meɪ/) is a lightweight JavaScript animation  
library with a simple, yet powerful API.  
It works with CSS properties, SVG, DOM attributes and JavaScript Objects.`,  
  },  
  {  
    title: "Three.js",  
    url: "https://threejs.org/",  
    description:  
      "Three.js is a cross-  
browser JavaScript library and application programming interface used to creat  
e and display animated 3D computer graphics in a web browser using WebGL.",  
  },  
  {  
    title: "Phaser.io",  
    url: "https://phaser.io/",  
    description:  
      "Phaser is a fast, free, and fun open source HTML5 game framework that o  
ffers WebGL and Canvas rendering across desktop and mobile web browsers. ",  
  },  
  {  
    title: "GSAP",  
    url: "https://greensock.com/gsap/",  
    description:  
      "GSAP is a JavaScript library for building high-  
performance animations that work in every major browser. Animate CSS, SVG, can  
vas, React, Vue, WebGL, colors, strings, motion paths, generic objects... anyt  
hing JavaScript can touch!",  
  },  
  {  
    title: "Mo.js",  
    url: "https://mojs.github.io/",  
    description:  
      "mo · js is a javascript motion graphics library that is a fast, retina  
ready, modular and open source. In comparison to other libraries, it have a di  
fferent syntax and code animation structure approach. The declarative API prov  
ides you a complete control over the animation, making it customizable with ea  
se.",  
  },  
  {  
    title: "Velocity.js",  
    url: "http://velocityjs.org/",  
  }  
]
```



```
description: `Velocity is an animation engine with the same API as jQuery's $.animate(). It works with and without jQuery. It's incredibly fast, and it features color animation, transforms, loops, easings, SVG support, and scrolling. It is the best of jQuery and CSS transitions combined.`,\n},\n{\n  title: "AniJS",\n  url: "https://anijs.github.io/",\n  description: "A Library to Raise your Web Design without Coding",\n},\n{\n  title: "vivus",\n  url: "https://maxwellito.github.io/vivus/",\n  description:\n    "Vivus is a lightweight JavaScript class (with no dependencies) that allows you to animate SVGs, giving them the appearance of being drawn. There are a variety of different animations available, as well as the option to create a custom script to draw your SVG in whatever way you like.",\n},\n{\n  title: "ScrollReveal",\n  url: "https://scrollrevealjs.org/",\n  description:\n    "ScrollReveal is a JavaScript library for easily animating elements as they enter/leave the viewport. It was designed to be robust and flexible, but hopefully you'll be surprised below at how easy it is to pick up.",\n},\n{\n  title: "Typed.js",\n  url: "https://mattboldt.com/demos/typed-js/",\n  description:\n    "Typed.js is a library that types. Enter in any string, and watch it type at the speed you've set, backspace what it's typed, and begin a new sentence for however many strings you've set.",\n},\n];
```

- Vous pouvez bien sûr repartir de la démonstration [js-demos/frontend/animejs/](https://mattboldt.com/demos/typed-js/). Néanmoins, afin d'apprendre à intégrer une librairie externe, nous vous conseillons de partir d'une page blanche et de suivre les instructions qui suivent.
- Apprenez à intégrer une librairie JS externe. Pour ce faire, utilisez la dernière version de la librairie anime.js sans CDN mais localement :



JavaScript & Node.js

Exercices

- Télécharger la dernière version :

<https://github.com/juliangarnier/anime/archive/master.zip>

La librairie se trouve dans le répertoire anime-master/lib

Vous pouvez utiliser la version de production : anime.min.js.

- Intégrer cette librairie à votre application :

```
<script src="anime.min.js"></script>
```

- Pour comprendre comment utiliser la librairie anime.js, inspirez-vous de la démo js-demos/frontend/animejs/ ainsi que de <https://animejs.com/documentation/>
- Pour faire bouger des mots, il est utile d'utiliser des éléments html dont le comportement d'affichage est celui attendu.

On pourrait utiliser des spans (comportement d'affichage de type « inline »), des paragraphes (comportement d'affichage de type « block »), des div...

Dans la démo, on a utilisé des div avec « inline-block » comme comportement d'affichage, via l'utilisation de la classe Bootstrap **d-inline-block**. Ça nous permet d'avoir des blocs qui prennent juste la taille de leur contenu.

NB : les différents types d'affichage :

https://www.w3schools.com/cssref/pr_class_display.asp



Le code de votre application web optionnelle devrait se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-exercices**).

