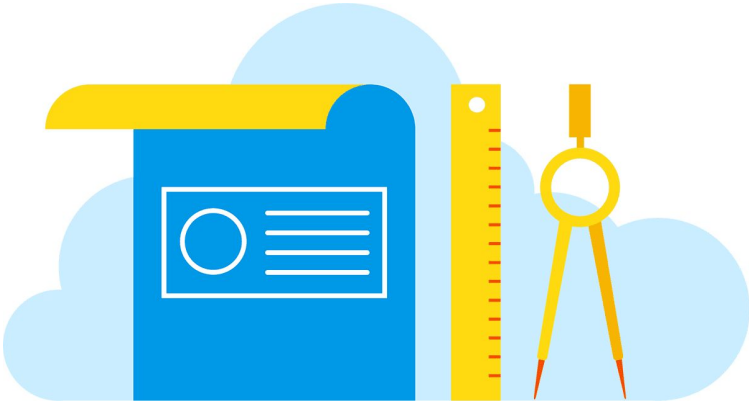


R312

Présentation et objectif

Durant ce premier module de reprise, nous reprendrons avec le CSS en utilisant les notions avancées du document CM, viendra ensuite une partie dynamique avec le chargement de données grâce à JavaScript et les “promises” ainsi que le format JSON pour le formatage des données.

L’objectif étant de créer une carte de jeu qui s’animera au survol et ensuite de pouvoir en générer plusieurs dynamiquement grâce au JavaScript et JSON

Supports nécessaires

1. Le CM
2. Les documents mis à votre disposition sur le cours R312 sur Moodle,
    - le fichier html de base
    - le fichier JSON
    - les images
    - la vidéo de démo

1. Reprenons en douceur avec CSS

- Recréez une carte de ce type, en utilisant **les mêmes informations** (vous en aurez besoin pour la suite du TP) :
  - Une image (format .png transparent)
  - Un nom de personnage
  - Une courte description
  - Des points de vie, de discrétion et d'action
- **Aucune** animation pour le moment

2. Animation de l'ombre portée

- Ajoutez une ombre portée au survol de votre carte (*card:hover { … }*)
- Utilisez pour cela la propriété **box-shadow**
- Une **transition** complètera votre animation qui, pour l'instant, change brutalement

3. Animation de l'image

Afin d'animer l'image vous utiliserez les propriétés suivantes :

- **transform** avec la modification d'échelle (**scale**) ainsi que la translation en Y (**translate**)
- Celle-ci s'appliquera également au survol de la carte (et non de l'image) : *card:hover ... { … }*

4. Animation des caractéristiques

Deux étapes sont nécessaires afin d'animer les caractéristiques du personnage :

1. Définir la **hauteur** du bloc ainsi que son **opacité** à 0 (ce dernier point est optionnel mais permet un transition plus jolie) : *height: 0; opacity: 0;*
2. **Au survol de la carte**, redéfinir la bonne hauteur (en px, pas de auto) ainsi que l'opacité à 1.

Allons plus loin avec la génération automatique de notre plateau de jeu

Préambule

Grâce au fichier json fourni et JavaScript nous allons créer nos cartes automatiquement

```
{ "nom" : "Woody", "vie": "25", "discretion" : "22" , "action"
: "40" , "photo" : "woody.png" , "info": "Les mains en l'air !
Le shérif Woody est la pour faire régner l ordre dans la
chambre d'Andy." },

{ "nom" : "Buzz l'éclair", "vie": "35", "discretion" : "15" ,
"action" : "70" , "photo" : "buzz.png" , "info": "Vers
l'infini et au-delà !" },
```

Fichier datastory.json, il contient toutes les informations utiles pour créer nos cartes de jeu.

Créez un dossier « exo4 » avec dedans le fichier datastory.json et le fichier exo04.html

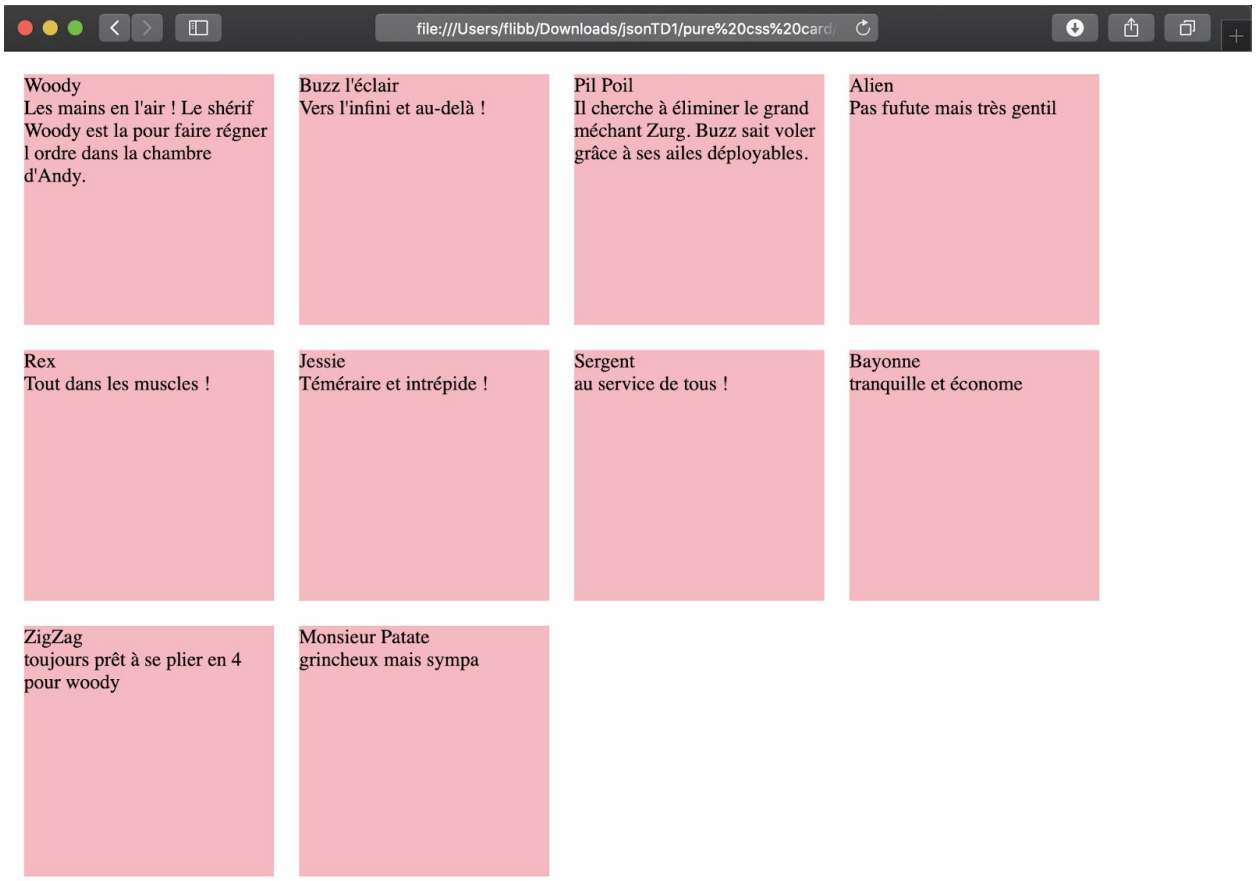
```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="UTF-8">
    <style type="text/css">

        ...

    </style>
</head>

<body>
<h1> demo </h1>
</body>
<script type="text/javascript">

...

</script>
</html>
```

Exercice 1 : gestion de la boucle pour afficher les éléments

Dans un premier temps nous allons créer la liste à partir d'un tableau JSON en « dur » dans le JavaScript :

```
<body>
    <script>
        var monTableau = [

            {
                "nom": "Woody","vie": "25","discretion": "22","action": "40","photo": "woody.png","info":
                "Les mains en l'air ! Le shérif Woody est la pour faire régner l ordre dans la chambre
                d'Andy."
            },

            {
                "nom": "Buzz l'éclair","vie": "35","discretion": "15","action": "70","photo": "buzz.png",
                "info": "Vers l'infini et au-delà !"
            },


        ];
        monTableau .... (function(element) {
            .... = '<div class="carte">' + element.nom + .... '</div>';
        });
    </script>
</body>
```

Complétez le source ci-dessus pour obtenir le résultat suivant (nom et description des personnages) :



Pour cela vous aurez besoin de la méthode ***« forEach() »*** pour traiter tous les élément du tableau et de « ***document.body.innerHTML*** » pour écrire dans le corps de la page (ou « ***document.getElementById('mazone').innerHTM***L » pour écrire dans un div en particulier via son id).
A vous de regarder la documentation en ligne de ces méthodes.

Exercice 2 : Chargement d'un fichier json grâce à javascript

Pour charger un json depuis un script javascript, il existe plusieurs méthode (Jquery, JavaScript natif, ...)

Nous allons utiliser ici la méthode **« fetch »** qui est maintenant incluse dans JavaScript.

Cette méthode utilise le concept des **« promises »** car le chargement d'un fichier externe ne doit pas être bloquant pour le reste du script (le chargement pouvant être long via une connexion distante, il ne faut pas que le reste du script attende pour être exécuté que tout le fichier soit chargé)
L'appel du fichier se passe comme cela :

```
<h1> demo </h1>
<div id="mazone">...</div>
<h2>fin</h2>
</body>
<script type="text/javascript">

console.log("demarrage") ;
fetch('mesdatas.json')
    .then( function(reponse)
            {
            console.log( "ça y est, l'entete du fichier est arrivée,..." ) ;
            console.log( "maintenant les data vont arriver");
            return reponse.json();
            }
        ) ;
console.log( "suite et fin du script") ;
</script>
```

Nous appelons ici ***« fetch »*** avec en paramètre le nom du fichier json à télécharger. (attention ceci doit se faire uniquement via un serveur web et non une simple ouverture du fichier html en local, vous devez donc avoir sur votre ordinateur un serveur web type MAMP, WAMP,... ou alors transférer vos fichier sur votre VPS)

Cette fonction renvoie ce que l'on appelle une ***« promise »*** : c'est à dire que la fonction passe tout de suite à l'instruction suivante (ici console.log(« suite et fin du script ») avant même que le fichier JSON est eu le temps de se télécharger (fonction asynchrone).
La méthode ***« then »*** a comme paramètre une fonction qui sera exécutée quand le fichier commencera à être téléchargé (en fait quand toute l'entête sera arrivée).

Du coup dans la console du navigateur on voit :

```
demarrage
suite et fin du script
ça y est, l'entete du fichier est arrivée,...
maintenant les data vont arriver
```

On voit bien ici que les messages « ça y est.... » arrive bien après le message « suite et fin du script »

La ligne « return reponse.json() ; » appelle la méthode json() qui charge le corps du fichier (cela peut prendre du temps si le fichier JSON est conséquent) et renvois aussi une « promise » qu'il faudra traiter comme tel, donc avec une méthode .then() qui se déclenchera quand toutes les data json seront arrivées cette fois.

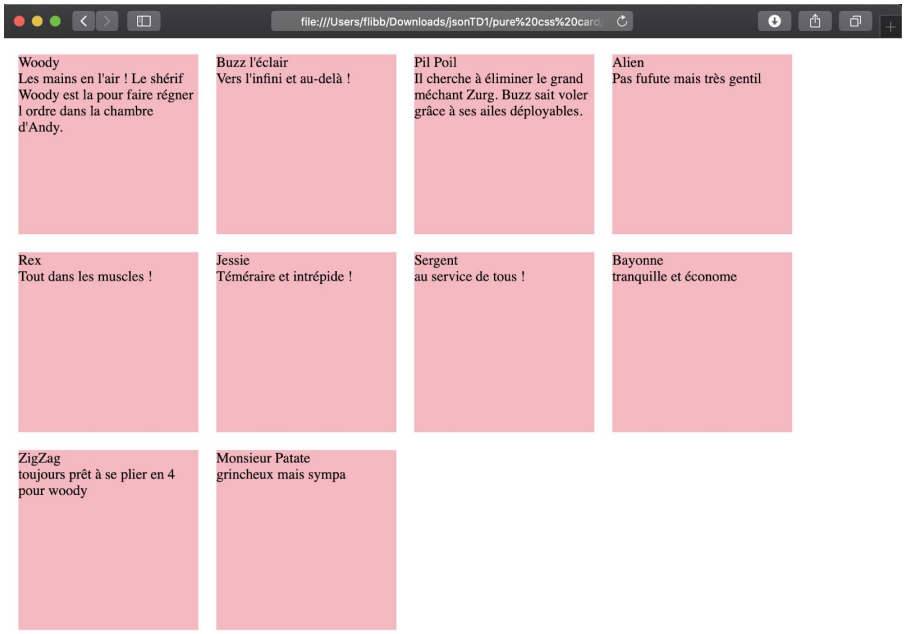
```
console.log("demarrage") ;
fetch('mesdatas.json')
    .then( function(reponse)
            {
            console.log( "ça y est, l'entete du fichier est arrivée,..." ) ;
            console.log( "maintenant les data vont arriver");
            return reponse.json();
            }
         )
        .then(function( reponse2 )
                {
                console.log("et ce coup-ci les datas sont arrivées! ")
                }
             ) ;

console.log( "suite et fin du script") ;
```

Dans la console on voit donc maintenant :

```
demarrage
suite et fin du script
ça y est, l'entete du fichier est arrivée,...
maintenant les data vont arriver
et ce coup-ci les datas sont arrivées!
```

Maintenant à la place du ***console.log(« et ce coup-ci les datas... »)*** , on peut placer notre code js permettant de traiter tout le tableau json reçu dans la variable ***« reponse2 »***.

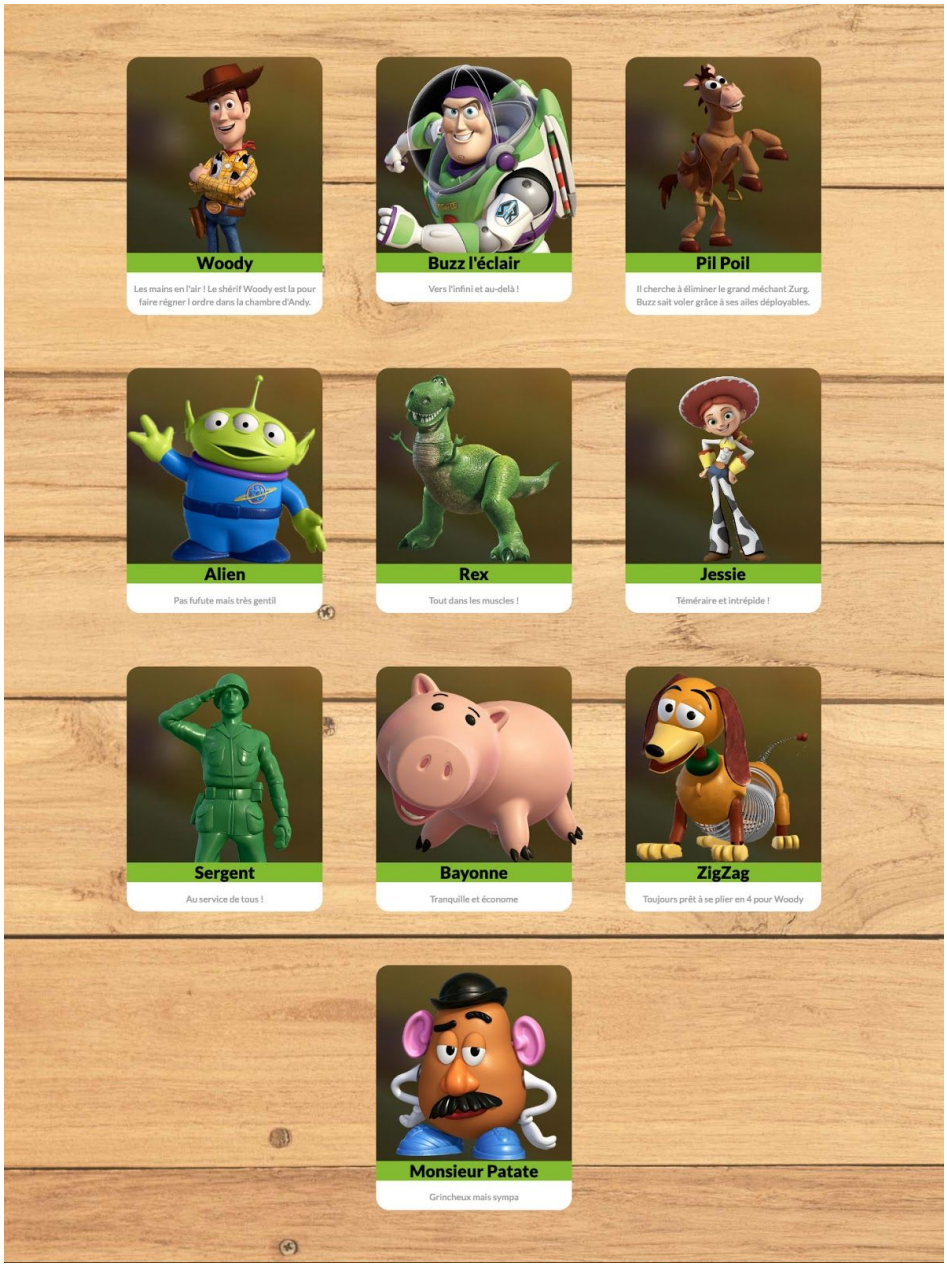
A vous de finaliser votre fichier pour obtenir le même résultat que l'exercice 1 :



Exercice 3, Vers l'infini...

Réalisez la même chose sauf que maintenant vous avez des informations supplémentaire dans le fichier json : l'image et les caractéristiques de chaque individu.

Modifiez votre source HTML pour afficher tous les personnages comme ci-dessous



Trop facile pour vous ?

Si vous avez terminé, vous pouvez maintenant faire un effet de retournement de carte.

Pour cela faites une carte avec un recto et un verso (qui lui sera inversé de 180°) puis animer le tout avec une rotation de 180°