Comprendre jQuery - Paint

Ce projet à pour but de vous faire mieux comprendre la manipulation basique du `DOM` en créant un programme ! Un peu comme **Microsoft Paint**.

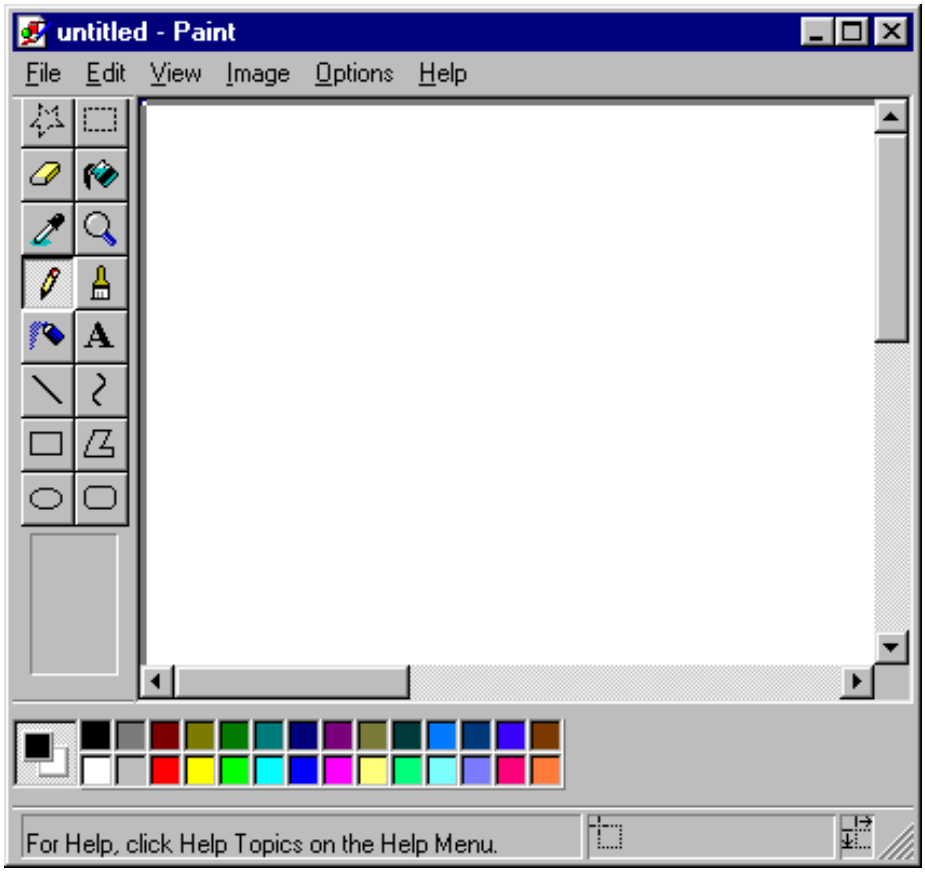

Figure 1: Microsoft Paint

Notre fichier `index.html` a 5400 divs à l'interieur. Les divs sont de petits carrés noirs. Ces carrés representent chaque pixels de notre programme. On ne se preoccupera pas du **HTML** & du **CSS**. Juste notre fichier nommé `app.js`.

Etape 1 - Ajouter la Classe

La première chose a faire à nos « pixels » est de changer la couleur. Ces petits gars seront les blocs de construction de notre œuvre d'art. Avec la v1 de notre application, nous allons juste faire des pixels noirs et blancs. Non pas parce

que nous ne pouvons pas avoir des couleurs, mais nous aurons besoin de créer quelques fonctionnalités pour la v2 ?

- Initialiser notre fichier `app.js` par le chargement du document quand il est prêt.

```
$(document).ready(function () {
// votre code
});
```

Tout votre code va se situer entre ces lignes de code.

- Faire en sorte que lorsque l'on clique sur un pixel, il devient blanc :
- Dans la notre fichier `index.html` chaque div a la classe de `box`
- Nous pouvons utiliser cette classe de `box` comme notre sélecteur. Cela nous donne un moyen d'interagir avec votre `DOM`

```
$('.box').on('click', function () {
  $('.box').addClass('white');
});
```

Notre fichier `style.css` a une classe nommée `white`. Tout ce qu'elle fait est de changer l'arrière-plan des boîtes du noir au blanc.

En théorie, cela changerait la boîte noire au clique en boîte `white`, mais que se passe-t-il quand on essaye ? Il semble que lorsque l'on clique sur une box, toutes nos boîtes se changent en blanc!

En effet, ils partagent tous la même classe, par le nom de la `box`. Nous sommes en train de dire à **jQuery** que tout se qui se refère à la classe `box` devrait aussi être classe `white`. Ceci est un problème commun dans **jQuery**. Maintenant, nous pourrions donner à chaque div de la liste des 5400 divs leur propre ID unique, mais j'ai une meilleure idée.

- Utilisez le mot-clé `this`:

```
$('.box').on('click', function () {
  $(this).addClass('white');
});
```

Le mot-clé `this` est vraiment commun en JavaScript et ça peut être confus. Dans ce cas, avec notre code **jQuery**, il fait référence à la case spécifique sur laquelle nous avons cliqué. Souvent, vous vous retrouvez dans des positions où vous avez besoin de changer une chose, dans un tableau de beaucoup. Si jamais vous vous retrouvez a changer chacun d'eux, expérimenter avec `this` un peu.

Etape 2 - Editions

Super, maintenant nous pouvons créer de belles œuvres d'art, de l'art en noir et blanc, mais de l'art néanmoins. Mais que se passe-t-il quand on fait une erreur ? Il n'y a aucun moyen de modifier notre œuvre. Nous allons corriger cela.

- Faire en sorte que lorsque l'on double clic, il change de nouveau au noir avec `removeClass` :

```
$('.box').on('click', function () {
  $(this).addClass('white');
});

$('.box').on('dblclick', function () {
  $(this).removeClass('white');
});
```

Etape 3 - Bouton Réinitialiser

Il semble que nous avons un bouton de réinitialisation. Faisons-le fonctionner. Rappelez-vous comment nous pourrions envoyer des commandes de couverture sur l'ensemble de nos pixels en utilisant le nom de la classe au lieu du pixel individuel ? Nous allons utiliser cette méthode pour créer notre bouton de réinitialisation.

- Soyez sur que vos boutons soit blancs :

```
$('#reset').on('click', function () {
  $('.box').removeClass('white');
});
```

Step 4 - Palette de Couleurs

Faisons que nos boutons de couleur fonctionnent!

- Créer une variable de couleur, et définir la valeur par défaut en blanc. Cette variable de couleur sera la classe nous ajouterons.

```
var color = 'white';
```

- Créer un événement de clic pour chaque couleur qui change la variable de couleur lors d'un clic :

```
$('#red').on('click', function () {
  color = 'red';
});

$('#blue').on('click', function () {
  color = 'blue';
});

$('#green').on('click', function () {
  color = 'green';
});

$('#yellow').on('click', function () {
  color = 'yellow';
});

$('#white').on('click', function () {
  color = 'white';
});
```

- Mettre à jour la fonctionnalité de classe `add` pour refléter notre variable de couleur plutôt que nos noms de classes réelles :

```
$('.box').on('click', function () {
  $(this).addClass(color);
});

$('.box').on('dblclick', function () {
  $(this).removeClass(color);
});

$('#reset').on('click', function () {
  $('.box').removeClass(color)
});
```

Une dernière chose qui est un peu foiré. Nos fonctions de réinitialisation et double-clic ne fonctionne pas pour le moment. Parce que nous avons placé la variable `color` à la place de l'action de la classe de suppression, il ne fera que respecter la couleur actuellement sélectionnée.

Dans la fonction jQuery de nous avons la possibilité de supprimer plusieurs classes de seulement séparant chaque classe, nous voulons supprimer par un espace, comme suit :

```
$('.box').removeClass('red blue green yellow white');
```

Ce qui est faisable, mais nous allons juste mettre ceux-là dans une variable de sorte que nous n'ayant pas besoin d'écrire autant de code :

```
var colors = 'white green red blue yellow'

$('.box').on('dblclick', function () {
  $(this).removeClass(colors);
});

$('#reset').on('click', function () {
  $('.box').removeClass(colors)
});
```

Et voilà c'est fini, on a fini de setup notre application. Libre à vous d'ajouter des fonctionnalités. Enjoy !