jQuery

Table des matières

1. Introduction

jQuery est une librairie, plus ou moins comme Bootstrap, qui permet d'écrire du code JavaScript de manière plus rapide / pratique. Il consiste en une série de fonctions pré-écrites que nous allons pouvoir utiliser telles-quelles, et possède sa propre syntaxe. Par exemple :

Sélectionner un élément HTML :

```
<div id="maDiv">
    Bonjour
</div>
<script src="https://code.jquery.com/jquery-3.7.1.js" integ
<script>
    // Javascript ici :)
    document.getElementById('maDiv').innerHTML = 'Salut';
    // jQuery ici :)
    $('#maDiv').html('Salut !')
</script>
```

Il fonctionne sur une logique de **sélecteurs**, qui sont les mêmes qu'en CSS, c'est-à-dire les balises html, le point **.** pour les classes et le dièse **#** pour les id. Et pour l'intégrer on utilise comme avec Bootstrap, un **CDN**.

https://releases.jquery.com/

2. Syntaxe

jQuery utilise le signe **$** en guise d'appel. À chaque fois que vous verrez le signe **$** dans du JavaScript, cela veut dire que c'est à jQuery qu'on fait appel. C'est là toute la subtilité de jQuery : ce n'est pas un nouveau langage en soi, bien qu'il possède sa propre syntaxe, mais il coexiste avec JavaScript !

Quelques exemples :

```
// Javascript ici :)
document.getElementById('maDiv').addEventListener('click', function() {
    alert('Hello');
});
// jQuery ici :)
$('#maDiv').click(function() {
    alert('Hello');
});
```

```
// Vérifier si un de mes élements a bien une classe :
document.getElementById('maDiv').classList.contains('maClasse');
$('#maDiv').hasClass('maClasse');
```

```
// Afficher ou masquer un élément :
document.getElementById('maDiv').style.display = 'none';
$('#maDiv').hide();
document.getElementById('maDiv').style.display = 'block';
$('#maDiv').show();
```

```
// Gérer le hover sur un élément :
$('#maDiv').hover(function(){
    // Faire un truc au survol
}, function() {
    // Faire un truc quand on sort
});
```

3. Fonctions utiles

- **.click**(function() { /* Action à effectuer * /}) ; permet de gérer un **évènement** (le click)
- **.change**(function() { /* Action à effectuer * /}) ; permet de gérer un **évènement** (le change)
- **.on('evenement',** function() { /* Action à effectuer * /}) sera la façon préférable de déclarer les eventListener. En effet, un évènement déclaré avec **on()** sera appliqué même si l'élément cible arrive **après** le chargement du DOM, ce qui ne sera pas le cas avec les fonctions évènements « classiques » comme **.click** ou **.change**
- **.html**(/* Contenu à écrire */) ; permet d'écrire dans l'élément cible. Si on ne lui passe pas d'argument, html() nous renvoie le contenu **déjà présent**.
- **.data**(/* nom de ma data */) ; permet d'accéder à l'attribut data que l'on a défini sur l'élément cible
- **.each**(function() { /* Action à effectuer * /}) ; permet d'initier une **boucle** sur un tableau. À chaque itération, on aura accès au sélecteur **$(this)**, qui correspondra à l'itération en cours
- **.addClass('classeAAjouter') / .removeClass('classeAEnlever') ;** permet d'ajouter ou d'enlever une classe à notre élément. Pratique pour rapidement ajouter ou enlever des propriétés CSS ! Si on ne passe aucun paramètre à **removeClass()**, alors **toutes** les classes sont supprimées
- **.toggleClass('classe')** permet de gérer l'ajout ou la suppression d'une classe sans devoir vérifier si elle existe déjà ou non.
- **.hover**(function() { /* Action à effectuer quand on entre dans la zone de survol * /}, function() { /* Action à effectuer quand on quitte la zone de survol * /})
- **.animate** permet de gérer une animation sur une propriété. Par exemple, on peut utiliser animate sur la propriété **scrollTop** afin d'emmener la barre de scroll sur un élément précis.
- **offset** permet d'accéder à la position d'un élément, avec offset().top et offset().left
- **height()** permet d'obtenir la taille d'un élément
- **width()** permet d'obtenir la largeur d'un évènement