jQuery

jQuery - Introduction

- jQuery est une bibliothèque créé en 2006 par John Resig et considérée aujourd'hui comme obsolète
- jQuery était parfois présentée comme un framework dans le sens ou elle regroupe plusieurs bibliothèques :
  - manipulation du DOM
  - gestion des événements
  - animations
  - requêtes AJAX
- Aujourd'hui on remplacerait jQuery par des frameworks plus modernes comme Angular ou bien un ensemble de bibliothèques comme React + axios + anime.js

jQuery - Compatibilité

- Avant Internet Explorer 9, Microsoft ne respectait par les normes web
- Par exemple addEventListener s'appelait attachEvent jusqu'à IE8
- jQuery dans sa version 1 incluait une couche de compatibilité qui fait qu'appeler la méthode de jQuery rendait le code compatible sur tout navigateur
- La compatibilité assurée sur tous les navigateurs + un large écosystème écosystème de plugin a fait de jQuery une référence pendant presque 10 ans

jQuery - Inspirant pour le web

- jQuery a introduit des concepts qui sont devenus des normes en JavaScript :
  - recherche dans le DOM par sélecteur CSS
  - méthodes prepend / append / before / after / remove sur un éléments
  - deferred (équivalent des promesses)

jQuery - Pourquoi le remplacer ?

- jQuery encapsule des méthodes natives, dégradant ainsi les performances
- Les méthodes ne font pas partie d'une norme, il faudra maintenir le code dans le futur
- Les normes ont évoluées et propose souvent le même fonctionnel sans avoir à charger de bibliothèques supplémentaires
- jQuery partage son code au site web via un objet global rendant impossible certaines optimisations comme le Tree Shaking
- Des clients AJAX plus modernes et plus légers existent (axios...)
- Les animations proposées sont peu performantes et dépassées
- Les plugins peuvent être remplacés aujourd'hui par des Web Components ou des composants de bibliothèques plus modernes quand on les utilise (Angular, React, Vue)
- La bibliothèque reçoit toujours des mise à jours mineures mais ne devrait plus recevoir de nouvelles fonctionnalités
- Dans la mesure du possible, il faudrait remplacer jQuery, sauf lorsqu'il est en dépendances d'autres bibliothèques (Datatables, Datepicker, Bootstrap) et que les Web Components (ou composants React / Angular / Vue) n'existent pas encore

jQuery - Helloworld

- jQuery peut être utilisé sous forme de balise script ou bien via un bundler comme webpack
- Par convention un objet jQuery est préfixé par *$* (ici *$input*, *$output*)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>DOM</title>
</head>
<body>
  <div>Name : <input class="input"></div>
  <p>Hello <span class="output"></span> !</div>
  <script src="https://code.jquery.com/jquery-3.4.1.js"></script>
  <script type="module">
    const $input = $('.input');
    const $output = $('.output');
    $input.on('input', () => $output.text($input.val()));
  </script>
</body>
</html>
```

jQuery - Variable Globale

- jQuery créé une variable globale appelée *jQuery* et un alias *$*
- En cas de conflit on peut désactiver la variable *$* avec la méthode *noConflict*

```
console.log($ === jQuery); // true
$.noConflict();
console.log($); // undefined
```

jQuery - Wrapper

- jQuery est un wrapper, il va encapsuler un élément du DOM

```
// avec jQuery
$(document.body).css('background-color', 'yellow');
// sans jQuery
document.body.style.backgroundColor = 'yellow';
```

- La sélection peut également directement se faire via un sélecteur CSS, pour ça jQuery utilise une bibliothèque appelée Sizzle : https://sizzlejs.com/
- Sizzle contient des sélecteurs non-standard comme *:odd* (au lieu de *:nth-child(odd)*) https://api.jquery.com/category/selectors/jquery-selector-extensions/

```
// sélecteurs CSS
$('body').css('background-color', 'yellow');
```

jQuery - Lectures / Ecriture

- Avec jQuery les méthodes d'écritures sont les mêmes que les méthodes de lectures
- L'écriture reçoit un paramètre supplémentaire : la valeur à écrire

```
// avec le DOM (propriété)
console.log(document.body.id); // test
document.body.id = 'test';

// avec le DOM (attribut HTML)
console.log(document.body.getAttribute('id')); // test
document.body.setAttribute('id', 'test');

// avec jQuery (propriété)
console.log($('body').prop('id')); // test
$('body').prop('id', 'test');

// avec jQuery (attribut HTML)
console.log($('body').attr('id')); // test
$('body').attr('id', 'test');
```

jQuery - Collections

- Lorsque l'objet jQuery contient plusieurs éléments, la lecture s'applique au premier, l'écriture à tous

```
<p>1</p>
<p>2</p>
<p>3</p>
<script src="https://code.jquery.com/jquery-3.4.1.js"></script>
<script type="module">
  console.log($('p').html()); // 1
  $('p').html('test'); // modifie les 3 paragraphes
</script>
```

jQuery - Chainage

- Une méthode d'écriture retourne l'objet sur lequel elle vient de s'appliquer permettant le chainage d'autres méthodes

```
<p>1</p>
<p>2</p>
<p>3</p>
<script src="https://code.jquery.com/jquery-3.4.1.js"></script>
<script type="module">
  $('p')
    .css('backgroundColor', 'yellow')
    .text('test')
    .mouseenter(() => {
      console.log('enter');
    })
    .mouseleave(() => {
      console.log('leave');
    });
</script>
```

jQuery - En savoir plus

- Le parcours d'un arbre
  https://api.jquery.com/category/traversing/
- La modification du DOM
  https://api.jquery.com/category/manipulation/
- La modification du Style
  https://api.jquery.com/category/css/
- Les événements
  https://api.jquery.com/category/events/
- Les formulaires
  https://api.jquery.com/category/forms/
- Les requêtes AJAX
  https://api.jquery.com/category/ajax/