



JavaScript

Manipulation de jQuery

Définition

- **jQuery** est une bibliothèque JavaScript libre, créer en 2005 par **John Resig**. Il permet de :
 - Manipuler l'arbre DOM des pages web
 - Changer/Ajouter des styles CSS, modifier des attributs, gérer les événements, etc.
 - Faire des requêtes AJAX.
 - Faire des animations

Installation

- jQuery ne nécessite aucune installation particulière. Il suffit d'intégrer le fichier **jQuery** dans la balise **script** de votre page HTML :
 - Après son téléchargement depuis le site officiel <http://jquery.com>
- Ou
 - À travers l'URL dans le CDN (Content Delivery Network)
 - <https://code.jquery.com/jquery-3.7.1.min.js>
 - <https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js>

Manipulation du DOM

- **Sélecteurs simple de jQuery :**

- En JavaScript, pour sélectionner un élément avec un ID "p1", On peut écrire :

```
document.getElementById("p1"); ou document.querySelector("#p1");
```

- alors qu'avec jQuery, il suffit d'écrire :

```
$("#p1");
```

Ou

```
jQuery("#p1");
```

Manipulation du DOM

- **Sélecteurs simples de jQuery :**
 - Syntaxe générale

```
$("Sélecteur CSS");
```

Ou

```
jQuery("Sélecteur CSS");
```

Manipulation du DOM

- **Sélecteurs avancés de jQuery:**

<code>\$(':button')</code>	Sélectionne les <code><button></code> et les <code><input type="button"></code>
<code>\$(':checkbox')</code>	Sélectionne les <code><input type="checkbox"></code>
<code>\$(':radio')</code>	Sélectionne les <code><input type="radio"></code>
<code>\$(':text')</code>	Sélectionne les <code><input type="text"></code>
<code>\$(':input')</code>	Sélectionne tous les champs de formulaire (input, textarea, select, button)
<code>\$(':checked')</code>	Coche les éléments cochés (checkbox, radio)
<code>\$(':selected')</code>	Sélectionne l' <code><option></code> sélectionnée dans un <code><select></code>
<code>\$(':eq(index)')</code>	Sélectionne l'élément à la position index (commence à 0)
<code>\$(':first')</code>	Sélectionne le premier élément du groupe (équivalent à <code>:eq(0)</code>)
<code>\$(':submit')</code>	Sélectionne les boutons de type submit (input ou button)

Manipulation du DOM

- **Méthode de sélection de jQuery:**

Méthode jQuery	Exemple	Interprétation
.has()	<code>\$("p").has("span")</code>	Sélectionne tous les <code><p></code> contenant un <code></code>
.filter()	<code>\$("p").filter(".souligne")</code>	Sélectionne les <code><p></code> ayant la classe <code>.souligne</code>
.not()	<code>\$("p").not(".souligne")</code>	Sélectionne les <code><p></code> ne possédant pas la classe <code>.souligne</code>
.first()	<code>\$("p").first()</code>	Sélectionne le premier paragraphe
.last()	<code>\$("p").last()</code>	Sélectionne le dernier paragraphe
.eq(index)	<code>\$("p").eq(1)</code>	Sélectionne le deuxième paragraphe (index 1)

Manipulation du DOM

- **Manipuler la classe d'un élément :**

Syntaxe	Exemple	Description
<code>.addClass()</code>	<code>\$("#p1").addClass("rouge");</code>	ajoute au paragraphe #p1 la classe "rouge"
<code>.removeClass()</code>	<code>\$("#p1").removeClass("rouge");</code>	retire au paragraphe #p1 la classe "rouge"
<code>.toggleClass()</code>	<code>\$("#p1").toggleClass("rouge");</code>	ajoute au paragraphe #p1 la classe "rouge" s'il ne l'avait pas et la lui retire s'il l'avait.
<code>.hasClass()</code>	<code>\$("#p1").hasClass("rouge");</code>	détermine si le paragraphe #p1 dispose de la classe "rouge" (renvoie True ou False).

Manipulation du DOM

- **Modifier un élément :**

méthode	exemple	description
.text()	<code>\$("#p1").text();</code>	renvoie le contenu textuel du paragraphe #p1 (sans les balises html).
.text()	<code>\$("#p1").text("Nouveau texte");</code>	change le contenu du paragraphe #p1 par "Nouveau texte".
.html()	<code>\$("#div1").html();</code>	renvoie la balise du #div1 et son contenu.
.html()	<code>\$("#div1").html("Nouveau paragraphe");</code>	change le contenu du #div1 par : " Nouveau paragraphe"

Manipulation du DOM

- **Modifier un élément :**

méthode	exemple	description
.attr()	<code>\$("#p1").attr('style');</code>	renvoie la valeur de l'attribut 'style' du paragraphe #p1.
.attr()	<code>\$(".lien1").attr("href" , "https://www.google.com/");</code>	change le contenu de l'attribut "href" du lien .lien1 par "https://www.google.com/"
.removeAttr()	<code>\$(".lien1").removeAttr("href");</code>	Supprime l'attribut "href" du lien .lien1
.val()	<code>\$("#input1").val();</code>	renvoie la valeur du champ #input1.
.val()	<code>\$("#input1").val("Mon texte");</code>	Mettre la valeur "Mon texte" dans champ #input1

Manipulation du DOM

- **Créer une balise :**
 - `let p3= $("<p>Texte du paragraphe 3</p>")` est équivalent à :
`let p3= document.createElement('p')`
`p3.textContent='Texte du paragraphe 3'`

Manipulation du DOM

Insertafter
after before

- **Ajouter une balise :**

Syntaxe	Exemple	Description	Équivalent à
.append()	<code>\$("#div1").append(p3)</code>	ajoute le paragraphe stocké dans p3 à la fin du #div1	<code>document.getElementById('div1').append(p3)</code>
.prepend()	<code>\$("#div1").prepend(p3)</code>	ajoute le paragraphe stocké dans p3 au début du #div1	<code>document.getElementById('div1').prepend(p3)</code>
.insertBefore	<code>\$(p3).insertBefore('#p1')</code>	Insère un p3 avant #p1	<code>Parent. insertBefore(p3, p1)</code>
.insertAfter	<code>\$(p3).insertAfter('#p1')</code>	Insère un p3 après #p1	<code>parentP1.insertBefore(p3, p1.nextSibling)</code>

Manipulation du DOM

- **Copier une balise :**

Syntaxe	Exemple	Description	Équivalent à
.clone()	<code>\$("#div1").clone()</code>	Copie l'élément #div1	<code>document.getElementById('div1').cloneNode(true)</code>

Manipulation du DOM

- **Déplacer une balise :**

Syntaxe	Exemple	Description	Équivalent à
<code>.replaceWith()</code>	<code>\$('#p1').replaceWith(\$('#p2'))</code>	remplace l'élément #p1 par l'élément #p2	<code>p1.replaceWith(p2)</code> Ou <code>parent.replaceChild(p1,p2)</code>

Manipulation du DOM

- **Supprimer une balise :**

Syntaxe	Exemple	Description	Équivalent à
<code>.remove()</code>	<code>\$("#div1").remove()</code>	supprime l'élément #div1	<code>div.remove()</code> Ou <code>parentDiv.removeChild(div)</code>

Manipulation du DOM

- **Notion de chaînage :**

Le chaînage permet d'exécuter plusieurs méthodes jQuery (l'une après l'autre) sur le même élément.

- **Exemple :**

```
$("<p></p>").addClass("Enrouge").text("Paragraphe ajouté à l'aide du  
chaînage").css("fontSize","30px").append(div1)
```

Ou

```
$("<p></p>")  
.addClass("Enrouge")  
.text("Paragraphe ajouté à l'aide du chaînage")  
.css("fontSize","30px")  
.append(div1)
```


Manipulation du DOM

- **Naviguer dans le DOM:**

syntaxe	Exemple	description
<code>.parent()</code>	<code>\$("#div1").parent() .css("border", "solid 3px red")</code>	encadre le parent de #div1
<code>.parents()</code>	<code>\$("#div1").parents() .css("border", "solid 3px red")</code>	encadre tous les ancêtres de #div1
<code>.children()</code>	<code>\$("#div1").children() .css("border", "solid 3px green")</code>	encadre les enfants directs de #div1
<code>.find()</code>	<code>\$("#div1").find() .css("border", "solid 3px green")</code>	encadre tous les descendants de #div1

Manipulation du DOM

- **Naviguer dans le DOM:**

syntaxe	exemple	description
<code>.siblings()</code>	<code>\$("#div1").siblings() .css("border", "solid 3px blue")</code>	encadre les frères de #div1
<code>next()</code>	<code>\$("#div1").next() .css("backgroundColor", "red")</code>	colorie le frère qui suit #div1
<code>prev()</code>	<code>\$("#div1").prev(). css("backgroundColor", "green")</code>	colorie le frère qui précèdent #div1

Evènements

- **La syntaxe de l'événement en jQuery :**

```
sélecteur.événement(function()  
{  
  // Code à exécuter  
})
```

Evènements

- **Exemple :**

```
$("#p").on(mouseover, function() {           /* code ici */  
})
```

```
$(document).ready(function() {               /* code ici */           })  
équivalent à  
document.addEventListener('DOMContentLoaded',  
() => { /* code ici */ }).
```

Exercice 1 :

Soit la structure HTML suivante :

```
<h1>Titre principal</h1>
<div>
  <p class='blue'>Paragraphe n°:1</p>
  <p id='p2'>Paragraphe n°:2</p>
  <p>Paragraphe n°:3</p>
  <p>Paragrahe sans attribut</p>
</div>
<p>Paragraphe n°:4</p>
```

En utilisant jQuery,

1. Copiez le paragraphe n°: 4 avant la fin de div
2. Déplacez le paragraphe n°: 2 après le titre principale
3. Supprimez le paragraphe n°: 3
4. Remplacez le paragraphe sans attribut par le paragraphe n°: 1

Exercice 2 :

On considère le code HTML ci-dessous :

```
<p class="vert">Paragraphe n°: 1</p>  
<p>Paragraphe n°: 2</p>  
<p class="vert">Paragraphe n°: 3</p>  
<p id="p4">Paragraphe n°: 4</p>
```

Ecrire un code jQuery qui permet de :

- 1- Ajouter la balise **<p> Paragraphe numéro 5 </p>** entre le paragraphe 2 et le paragraphe 3
- 2- Ajouter une bordure et une couleur de l'arrière plan au paragraphe n°: 2
- 3- Colorier les paragraphes possédant l'attribut class='vert' en vert
- 4- Ajouter au paragraphe 4 l'attribut class= 'rouge'
- 5- Supprimer l'attribut id= 'p4' du paragraphe 4

Exercice 3 :

On considère le code suivant:

```
<select id="genres">  
  <option value="rock">Rock</option>  
  <option value="blues" selected>Blues</option>  
</select>
```

Utilisez jquery pour :

1. Afficher la valeur et le texte de l'option sélectionnée.
2. Ajouter l'option : `<option> value = "classic"> Classic </option>` après l'option **Rock**.
3. Rendre l'option **classic** sélectionnée par défaut.

Exercice 4

Ecrivez le code du formulaire ci-contre en jQuery, sachant que :

- Après le chargement de la page ou lorsqu'on clique sur le bouton **nouveau calcul**, les deux premiers champs génèrent des nombres entre 1 et 10
- Quand on quitte le troisième champ, la réponse affiche juste ou fausse

Que vaut le calcul suivant ?

x =

la réponse est :



Découvrir AJAX

Introduction

- **AJAX** (Asynchronous JavaScript and XML) est une technique basée sur l'objet **XMLHttpRequest** de JavaScript.
- L'objet **XMLHttpRequest** permet d'envoyer des requêtes HTTP au serveur, de recevoir des réponses et de mettre à jour la page Web de façon transparente pour l'utilisateur en mode asynchrone.

Fonctionnement d'AJAX

- Méthodes d'objet XMLHttpRequest

Méthode	Description
new XMLHttpRequest()	Créer un nouvel objet XMLHttpRequest
open(method, url, async, user, psw)	Spécifier la requête : <ul style="list-style-type: none">• method: GET ou POST• url: emplacement du fichier• async: true (asynchrone, c'est à dire JavaScript n'a pas à attendre la réponse du serveur) ou false (synchrone)• user: nom d'utilisateur (optionnel)• psw: mot de passe (optionnel)
send()	Envoyer la requête au serveur (utilisé dans les requêtes GET)
send(string)	Envoyer la requête au serveur (utilisé dans les requêtes POST)

Fonctionnement d'AJAX

- Propriétés d'objet XMLHttpRequest

Propriété	Description
readystatechange	Evènement qui se déclenche lorsque la propriété readyState a été changée
readyState	Contient l'état de XMLHttpRequest.
responseText	Renvoie les données de réponse sous forme de chaîne

Fonctionnement d'AJAX

- Les états de la propriété readyState

CONSTANTE	VALEUR	Description
UNSENT	0	L'objet xhr a été créé, mais pas initialisé (la méthode open() n'a pas encore été appelée).
OPENED	1	Connexion au serveur établie. La méthode open() a été appelée, mais la requête n'a pas encore été envoyée par la méthode send().
HEADERS_RECEIVED	2	La méthode send() a été appelée et toutes les informations ont été envoyées au serveur.
LOADING	3	Le serveur traite les informations et a commencé à renvoyer les données.
DONE	4	Toutes les données ont été réceptionnées.

Fonctionnement d'AJAX

- Propriétés d'objet XMLHttpRequest

Propriété	Description
status	Renvoie le numéro d'état d'une requête 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Renvoie le texte d'état (par exemple "OK" ou "Not Found")

Fonctionnement d'AJAX

- Exemple :

```
<p>Charger les données d'un fichier texte nommé file.txt du serveur</p>
<input type="button" value="Charger">
<p id="fileContent"> <span>Aucun fichier chargé</span> </p>

<script>
  let input = document.querySelector("input")
  input.addEventListener("click", () => {
    let xhr = new XMLHttpRequest(); //Création de l'objet xhr
    xhr.open('GET', "file.txt", true) //Connexion au serveur
    xhr.send() //Envoi de la requête
    xhr.addEventListener('readystatechange', () => {
      if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
        // si la requête est terminée avec succès, responseText reçoit la réponse
        document.getElementById("fileContent").textContent = xhr.responseText
      }
    })
  })
</script>
```

Implémentation d'AJAX via jQuery

- La méthode `$.ajax()` permet d'envoyer des requêtes HTTP asynchrones au serveur, afin d'échanger des données (en lecture ou en écriture) sans recharger la page."

- Syntaxe :

`$.ajax(options)`

Implémentation d'AJAX via jQuery

- Certaines options :

Options	Description
url	Une chaîne contenant l'URL à laquelle la demande est envoyée.
type	Un type de requête http : POST ou GET. La valeur par défaut est GET.
data	Une donnée à envoyer au serveur.
dataType	Le type de données attendues du serveur.
success	Une fonction de rappel à exécuter lorsque la requête Ajax réussit.
error	Une fonction de rappel à exécuter lorsque la requête échoue.

Implémentation d'AJAX via jQuery

- Exemple :

```
$.ajax('file1.txt',  
{  
  dataType: 'text', // type des données de la réponse  
  
  success: function (data,status,xhr) { // fonction en cas de succès  
    $('#fileContent').html("Data : "+data+ " <br> status : "+ status + "<br> xhr : "+ xhr.readyState);  
  },  
  error: function (jqXHR, textStatus, errorMessage) { // cas d'erreur  
    $('#fileContent').html('Error : ' + errorMessage + '<br>textStatus : '+textStatus+ "<br> xhr : "+ jqXHR.readyState);  
  }  
});
```

Data : Bonjour tout le monde

status : success

xhr : 4

cas d'une requête terminée avec succès

Error : Not Found

textStatus : error

xhr : 4

cas d'une requête terminée avec erreur

\$.get()

- Méthode \$.get() envoie une requête http GET asynchrone au serveur et récupère les données.
- Syntaxe :
\$.get(
 - 'fichier', // Le fichier à appeler sur serveur.
 - 'text', // Format des données retournées par le serveur.
 - fonction, // Le nom de la fonction à appeler);

\$.get()

- Exemple :

```
$.get('file1.txt','text',  
function (data,status,xhr) { // fonction en cas de succès  
$('#fileContent').html("Data : "+data+ " <br> status : "+ status + "<br> xhr : "+ xhr.readyState);  
})
```

\$.post()

- La méthode \$.post() envoie une **requête POST** à l'URL spécifiée avec éventuellement des **données**, puis exécute une fonction de **rappel (callback)** si la réponse est réussie.
- Syntaxe :
- \$.post(
 - 'URL', // Le fichier à appeler sur le serveur.
 - Données, // les données à envoient au serveur
 - fonction, //fonction de retour);

\$.post()

- Exemple:

```
<form id="contactForm">
  <label>Nom : <input type="text" id="nom" required></label><br>
  <label>Email : <input type="email" id="email" required></label><br>
  <button type="submit">Envoyer</button>
</form>

<div id="message"></div>
```

```
<script>
  $("#contactForm").on("submit", function (e) {
    e.preventDefault(); // Empêche l'envoi classique

    // Récupération manuelle des champs
    let nom = $("#nom").val();
    let email = $("#email").val();

    // Construction d'un objet JavaScript
    let donnees = {
      nom: nom,
      email: email
    };

    // Envoi avec $.post()
    $.post("traitement.php", donnees,
      (reponse) => $("#message").html("<p>" + reponse + "</p>"));
  });
</script>
```

```
<?php
$nom = $_POST['nom'] ?? '';
$email = $_POST['email'] ?? '';

if ($nom && $email) {
  echo "Bonjour $nom, nous avons bien reçu votre email : $email.";
} else {
  echo "Veuillez remplir tous les champs.";
}
?>
```

\$.load()

- La méthode \$.load() permet de charger du contenu HTML ou texte à partir d'un serveur et de l'ajouter dans un élément DOM.
- Exemple :

```
$('#fileContent').load("file1.txt")
```